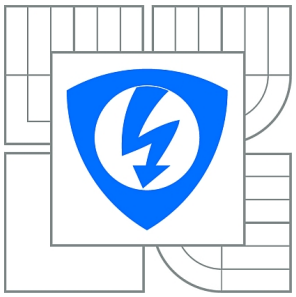
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

ŘÍZENÍ POHYBU STANICE V SIMULAČNÍM PROSTŘEDÍ OPNET MODELER PODLE MAPOVÉHO PODKLADU

MAP-BASED MOVEMENT CONTROL OF MOBILE STATION IN OPNET MODELER SIMULATION ENVIRONMENT

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

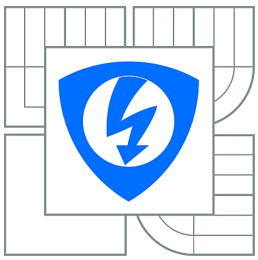
AUTOR PRÁCE — Bc. PETER JAKÚBEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ HOŠEK

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Peter Jakúbek	***ID:***	98113
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Řízení pohybu stanice v simulačním prostředí OPNET Modeler podle mapového podkladu

POKYNY PRO VYPRACOVÁNÍ:

V rámci diplomové práce se zaměřte na problematiku řízení pohybu mobilní stanice v simulačním prostředí OPNET Modeler. Vytvořte zjednodušený mapový podklad, který bude obsahovat více typů objektů definovaných různými barvami. Tento podklad zpracujte nástrojem MATLAB do takové podoby, aby byl použitelný v prostředí OPNET Modeler pro řízení pohybu mobilní stanice v rámci bezdrátové sítě. Následně pomocí nástroje MATLAB Engine propojte simulační prostředí OPNET Modeler s programem MATLAB a celý proces zautomatizujte tak, aby se veškeré jeho dílčí části spouštěly v rámci jedné simulace řízené prostředím OPNET Modeler. Všechny dosažené poznatky a výsledky podrobně zdokumentujte v závěrečné práci.

DOPORUČENÁ LITERATURA:

[1] ILYAS, M.: The Handbook of Ad Hoc Wireless Networks. Boca Raton: CRC Press, 2003, ISBN: 0-8493-1332-5.
[2] MOORE, H.: MATLAB for Engineers. New York: Prentice Hall, 2008, ISBN: 978-0136044222.
[3] OPNET Technologies, OPNET Modeler Product Documentation Release 16.0, 2010.

Termín zadání:	7.2.2011	***Termín odevzdání:***	26.5.2011

Vedoucí práce: Ing. Jiří Hošek

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Diplomová práca sa zaoberá problematikou riadenia pohybu mobilnej stanice v simulačnom prostredí OPNET Modeler (OM). V práci sú zhrnuté a popísané stávajúce možnosti riadenia pohybu bezdrôtovej mobilnej stanice.

Značná časť je venovaná riadeniu pohybu stanice pomocou externého súboru obsahujúceho trajektóriu pohybu. Pri tomto type riadenia sa jedná a spojitý pohyb stanice. Pohyb je riadený simulačným jadrom OM. Externé súbory určené pre riadenie pohybu stanice je možné vytvoriť ručne, alebo ich použiť na zápis dát priamo z OM pre uloženie vygenerovanej náhodnej trasy pre neskoršie použitie. Pre ručné vytváranie trajektórii je zvolený nástroj MATLAB, ktorý spracováva vopred vytvorený mapový podklad vo forme vstupného obrázka do takej podoby, aby bol výstup z nástroja použiteľný pre riadenie pohybu stanice v OM. Pre zautomatizovanie procesu vytvorenia trajektórie a následného riadenia pohybu stanice podľa nej je použitá knihovňa MATLAB Engine. Umožňuje volanie MATLAB funkcií z jazyka C/C++, na ktorom je založený OM.

Pre automatizovaný proces riadenia pohybu stanice je použitá priama manipulácia pozície stanice, ktorá je alternatívou riadenia pohybu stanice podľa súboru s trajektóriou. Priama manipulácia pozície stanice poskytuje nespojitý pohyb stanice. O riadenie sa stará užívateľom vytvorený proces, ktorý používa diskrétne hodnoty trajektórie.

KĽÚČOVÉ SLOVÁ

Bezdrôtová stanica, MATLAB Engine, mobilita, OPNET Modeler, trajektória pohybu

ABSTRACT

Diploma thesis focuses on a motion control of a mobile station in the simulation environment OPNET Modeler (OM). The thesis is summarized and described the possibilities of controlling the motion of a wireless mobile station.

A significant part concentrates on the motion controlling of the station via an external file containing the path of movement. In this type of governance is a continuous movement of the station. The movement is driven by a simulation engine OM. External files designed for a motion control of the station can be created manually or used to write a data directly from OM to store the random generated routes for later use. For the manual creation of the trajectory is chosen the MATLAB tool that handles pre-created maps in the form of the input image into a form that the output of the tool can be used for motion control station in OM. To automate the process of creation and the subsequent trajectory motion control station is used by her library MATLAB Engine. It allows to call MATLAB functions from the language C/C ++ on which is based OM.

For automated process of the movement control of the station is used a direct manipulation with the station position which is an alternative of the motion control station according to a set of the trajectory. The direct manipulation station position provides discrete movement of the station. The user's created process which uses the discrete trajectory values manage the control.

KEYWORDS

Wireless station, MATLAB Engine, mobility, OPNET Modeler, trajectory movement

JAKÚBEK, P. Řízení pohybu stanice v simulačním prostředí OPNET Modeler podle mapového podkladu . Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 69 s. Vedoucí diplomové práce Ing. Jiří Hošek.

Prohlášení

Prohlašuji, že svoji diplomovou práci na téma Řízení pohybu stanice v simulačním prostředí OPNET Modeler podle mapového podkladu jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 19. 5. 2011 .....

podpis autora

POĎAKOVANIE

Rád by som sa poďakoval Ing. Jiřímu Hoškovi a celému tímu, ktorý sa zaoberá simulačným prostredím OPNET Modeler za konzultácie, pripomienky a návrhy, ktoré mi ochotne poskytli v dobe vypracovávania tejto práce. Tiež sa chcem poďakovať mojej rodine za podporu počas celého obdobia môjho doterajšieho štúdia.

Obsah

Zoznam obrázkov:

1 Úvod

Bezdrôtové počítačové siete v dnešnej dobe stále pomaly a nezadržateľne vstupujú nie len do života počítačových odborníkov, technikov, manažérov, ekonómov, ale aj do sveta bežných užívateľov.

Potreba rýchlej a spoľahlivej komunikácie je azda jednou z najdôležitejších potrieb modernej spoločnosti. Počítačové siete tu s nami sú a berieme ich ako samozrejmosť so všetkými ich výhodami a obmedzeniami. Pri návrhu a vývoji nie len bezdrôtových, ale aj iných sietí je potrebná analýza, podľa ktorej sa postupuje pri vývoji technológie. Na toto slúžia simulátory, ktoré sú cennými nástrojmi nie len pri vývoji novej technológie, ale aj vo vzdelávacích odvetviach.

Jedným z typov počítačových sietí sú MANET (Mobile ad-hoc Networks) siete. Keďže sa jedná o mobilné siete, dôležitým faktorom je riadenie pohybu staníc počas simulácie. Z tohto dôvodu sa stáva riadenie pohybu staníc alebo celých podsietí kľúčovým pre simuláciu komunikačných sietí. Bezdrôtová komunikácia umožňuje pri zachovaní funkcionality staníc a podsietí meniť ich polohu založenú na preddefinovaných trajektóriach, orbitách a náhodne vybraných cestách.

2 MANET siete

MANET siete [1,2], niekedy nazývané mobilné mesh siete, sú mobilné ad-hoc siete, ktoré si samy konfigurujú sieť mobilných zariadení prepojených pomocou bezdrôtovej technológie. Každé zariadenie v MANET je voľné pohybom nezávisle na smere. Z tohto dôvodu bude meniť často svoje pripojenie k ostatným zariadeniam. Každé zariadenie zasiela dáta mimo svoju sieť a preto sa musí chovať podobne ako smerovač. Hlavnou výzvou pri budovaní MANET je vybavenie každého zariadenia tak, aby bolo schopné súvislého spracovania informácií potrebných na správne určenie cesty pre prenos dát. Takéto siete môžu operovať medzi sebou, alebo môžu byť pripojené do Internetu.

MANET siete sú druh bezdrôtových ad-hoc sietí, ktoré majú zvyčajne sieťové prostredie schopné smerovania na vrchu spojovej vrstvy ad-hoc sietí. Vzrast prenosných počítačov a technológie Wi-Fi (Wireless Fidelity) urobil z MANET sietí populárnu tému pre výskum od polovice deväťdesiatych rokov. Typy MANET sietí:

- **VANET** (Vehicular ad-hoc Networks) – sú používané pri komunikácii dopravných prostriedkov medzi sebou a medzi dopravnými prostriedkami a zariadeniami na okraji cesty.

- **InVANET** (Intelligent Vehicular ad-hoc Networks) – sú druhom s podporou umelej inteligencie, ktorá pomáha dopravným prostriedkom správať sa inteligentne počas kolízie vozidiel, nehôd, riadenia pod vplyvom alkoholu a podobne.

- **iMANET** (Internet Based Mobile ad-hoc Networks) – sú ad-hoc siete, ktoré prepájajú mobilné stanice s pevnými stanicami, ktoré reprezentujú bránu do internetu. V tomto druhu sietí sa klasický ad-hoc smerovací algoritmus neaplikuje priamo.

3 Možnosti riadenia pohybu stanice v simulačnom prostredí OPNET Modeler

Vzdialenosť je dôležitý parameter pri implementovaní rádiovo orientovaných technológií. Riadenie pohybu staníc je kľúčovým pre simuláciu komunikačných sietí. Pri bezdrôtovej komunikácii sa mení poloha staníc, ktorá je založená na vopred definovaných trajektóriach alebo náhodne vybraných cestách. Poloha mobilnej stanice je automaticky aktualizovaná počas simulácie a teda sa modeluje súvislý pohyb.

V OM (OPNET Modeler) je možné modelovať pohyb definovaním trajektórii a orbít, alebo priamou manipuláciou s hodnotami pozícii [3]. Pri simulovaní môže byť pre každý jednotlivý prípad mobilnej stanice použitý iba jeden z týchto mechanizmov. Ak nieje definovaná trajektória pohybu, je možné priamo meniť pozíciu mobilnej stanice pomocou nejakého procesu. Stanica musí začať jej pohyb v rámci svojej podsiete a potom sa môže pohybovať do inej podsiete.

V OM je niekoľko možností pre riadenie pohybu stanice. Sú to:

- **Konfigurácia náhodnej mobility –** jedná sa o implementáciu riadenia pohybu stanice na základe konfigurácie náhodnej mobility pomocou objektu „Moblity config“. Pomocou tohto objektu je možné vytvárať profily, ktoré majú nastavené rôzne parametre náhodnej mobility a následne ich priradiť mobilným staniciam.

- **Konfigurácia trajektórie –** doplnenie možnosti riadenia pohybu stanice pomocou náhodnej mobility. Konfigurácia náhodnej mobility je doplnená o možnosť riadenia z externého vstupu, vďaka čomu je stanica riadená priamo užívateľom. Nejedná sa teda o náhodnú trajektóriu ale o trajektóriu definovanú užívateľom.

- **Priama manipulácie pozície stanice** – priame manipulovanie pozície stanice pomocou procesu. Vzhľadom na diskrétne hodnoty pozície staníc sa jedná o nespojitý pohyb.

- **Orbity** – vytváranie orbít pomocou externého nástroja a ich následná implementácia do OM.

Práca sa venuje možnostiam konfigurácie náhodnej mobility, konfigurácia trajektórie a priama manipulácia pozície stanice. Nástroj MATLAB bol použitý na spracovanie mapového podkladu v podobe bitmapového obrázka a vygenerovanie súradníc pre ďalšie spracovanie. Súradnice sú ďalej spracované do súboru s trajektóriou, alebo priamo predávané do OM pomocou knižnice MATLAB Engine. Súbor s trajektóriou je ďalej implementovaný do nástroja OM pre reprezentáciu riadenia pohybu stanice pomocou konfigurácie trajektórie na základe spracovaného mapového podkladu. Priame predávanie hodnôt reprezentuje automatizovaný systém s priamou manipuláciou pozície stanice.

3.1 Konfigurácia náhodnej mobility

Trajektórie a orbity presne určujú deterministické cesty pre mobilné objekty [3]. Pre modelovanie náhodnej mobility je možné použiť modul náhodnej mobility. Náhodná mobilita

umožňuje definovať obdĺžnikovú oblasť v ktorom sa stanica bude pohybovať počas simulácie. Túto oblasť je možné definovať špecifikovaním súradníc X a Y, alebo použitím mobilnej domény. Počas simulácie mení stanica náhodne svoj cieľ a presúva sa k nemu špecifikovanou alebo náhodne určenou rýchlosťou. Na základe dosiahnutia svojho cieľa sa stanica zastaví a nakonfiguruje časový úsek predtým ako zopakuje proces výberu ďalšieho cieľa. Objekt mobilnej konfigurácie si drží parametre náhodnej mobility v profiloch pre možnosť ich opätovného použitia. Profily sú aplikované na objekty cez menu možnosti. Ak bude aplikovaný jeden profil viacerým objektom, ich pohyb bude rozdielny, pretože sa ciele pre každú stanicu vyberajú náhodne a nezávisle. Postup pre konfiguráciu náhodnej mobility je následovný:

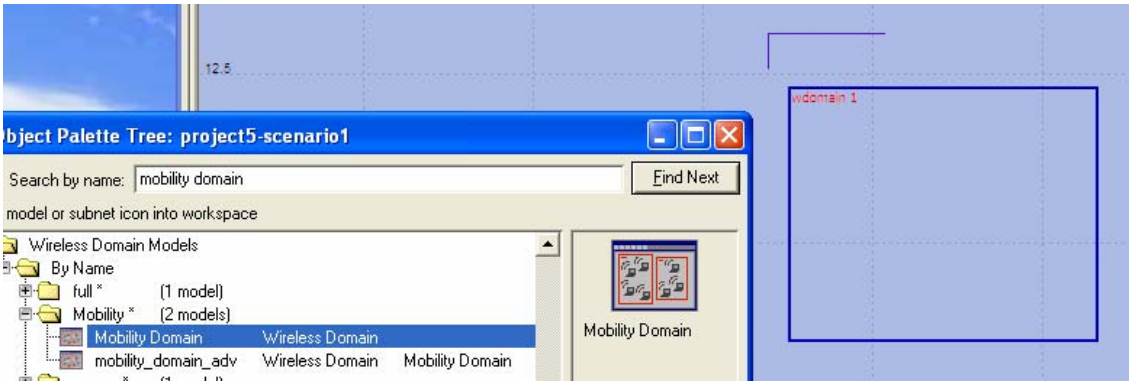
1. Definovanie mobilnej domény, ak je potrebné
2. Definovanie mobilného profilu
3. Aplikovanie profilu mobilnej stanici.

3.1.1 Definovanie mobilnej domény

Nastaviť náhodnú mobilitu je možné za pomoci použitia viacnásobných profilov pomocou objektu „Mobile Config“ [3]. Každá definícia profilu si vyžaduje špecifikovať oblasť pre definovanie rozsahu použiteľnosti tohto mobilného profilu. Táto oblasť môže byť špecifikovaná dvoma spôsobmi a to explicitne, určením minimálnych a maximálnych X a Y súradníc, alebo použitím objektu „Mobility Domain“. Postup vytvorenia ohraničeného poľa pre špecifikovanie rozsahu použiteľnosti jednotlivých profilov je nasledovný:

1. Z manet palety objektov vyberieme objekt „Mobility Domain“.
2. V pracovnom priestore projektu klikneme na ľavý horný roh oblasti, ktorú chceme definovať
3. Pretiahneme kurzor k pravému dolnému rohu želanej oblasti a klikneme pre ukončenie definovania oblasti. Mobilná doména sa objaví ako modrý obdĺžnik v pracovnom priestore. Implicitný názov sa objaví v ľavom hornom rohu červenou farbou.
4. Opakovaním krokov 2 a 3 môžeme definovať ďalšiu oblasť, alebo ukončiť operáciu pravým kliknutím myši.
5. Veľkosť mobilnej domény je možné upraviť premiestnením rohových bodov podľa potreby.
6. Nastavíme atribút názvu v nastaveniach atribútov mobilnej domény.

Na obr. 3.1 je názorná ukážka definovanej mobilnej domény s názvom „wdomain 1“.



Obr. 3.1: Mobilná doména

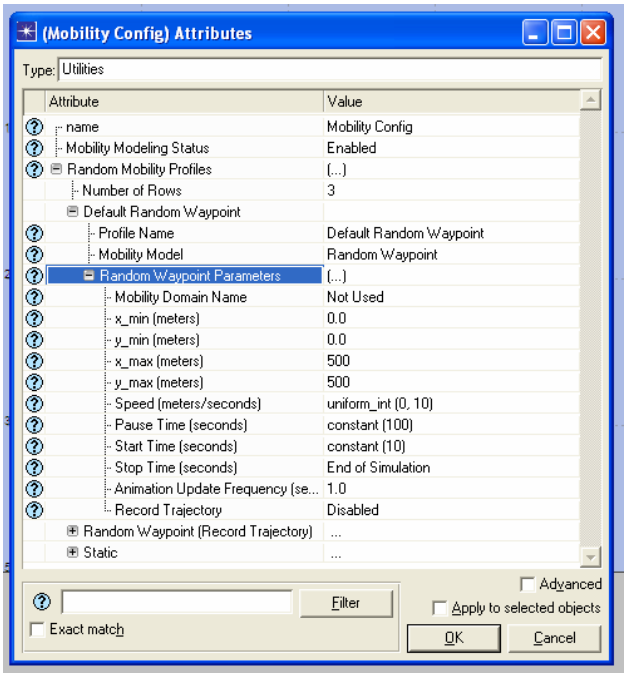
3.1.2 Konfigurácia profilov náhodnej mobility

Postup pre konfiguráciu profilov náhodnej mobility je nasledovný [3]:

1. Pretiahneme objekt „Mobility Config“ z manet palety objektov do pracovného priestoru OM, viď obr. 3.2. V jednom objekte môže byť uložených niekoľko profilov.

Obr. 3.2: Objekt „Mobility Config“

2. Pravým kliknutím na objekt otvoríme nastavenia atribútov „Mobility Config“ .
3. Editovaním atribútov profilu môžeme meniť parametre jedného z profilov. Nový profil môžeme vytvoriť pridaním riadku a nastavením atribútu mena.
4. Editujeme atribút „Random Waypoint Parameters“ profilu, ktorý chceme nastavovať.
5. Špecifikujeme oblasť, pre ktorú má byť profil použitý. Do atribútu „Mobility Domain Name“ nastavíme meno mobilnej domény, alebo explicitne určíme minimálne a maximálne X a Y súradnice. V prípade explicitného nastavenia súradníc musí byť atribút „Mobility Domain Name“ nastavený na hodnotu „Not Used“.
6. Ďalšie nastaviteľné atribúty a ich počiatočné hodnoty z profilu „Random Waypoint Parameters“ sú: „Speed“ = „uniform_int(0,10)“, „Pause Time“ = „constant(100)“, „Start Time“ = „constant(10)“ a „Stop Time“ = „End of Simulation“. Ak sú atribúty „Speed“ alebo „Start Time“ nastavené na hodnotu „None“, nenastane žiadný pohyb.
7. Ak chceme zaznamenávať náhodnú trasu generovanú týmto profilom pre neskoršie použitie, nastavíme atribút „Record Trajectory“ na hodnotu „Enabled“.
8. Parametre uložíme pomocou tlačidla „OK“. Vzorové nastavenia s počiatočnými hodnotami profilu „Random Waypoint Parameters“ sú zobrazené na obr. 3.3.



Obr. 3.3: Nastavenia objektu „Mobility Config“

3.1.3 Nastavenie profilov mobility mobilným staniciam

Stanici nastavíme mobilný profil nasledovne:

1. Označíme jednu alebo viac staníc, na ktoré chceme aplikovať jednotlivé mobilné profily.
2. Zvolíme „Topology > Random Mobility > Set Mobility Profile“, z ponuky vyberieme názov mobilného profilu pre označené stanice a potvrdíme tlačidlom „OK“

3.1.4 Vytváranie súboru *.trj s náhodnou trajektóriou

V niektorých prípadoch je potrebné zopakovať náhodnú cestu mobilnej stanice [3]. To môže byť užitočné pre porovnávanie simulačných výsledkov. Náhodná mobilita umožňuje zaznamenávať cestu generovanú počas simulácie a následne ju aplikovať ako trajektóriu pri ďalšej simulácii. Ak je atribút „Record Trajectory“ nastavený na hodnotu „Enabled“, OM generuje súbor s trajektóriou pre každú stanicu používajúcu tento profil počas simulácie. Každý súbor s trajektóriou obsahuje zoznam súradníc, po ktorých sa stanica pohybovala. Súbor má názov vo formáte „<project-scenario>-<node-name>.trj” a je umiestnený v adresári projektu. Súbor s trajektóriou je možné vytvoriť aj ručne a priradiť ho jednej alebo viacerým staniciam.

3.2 Konfigurácia trajektórii

Trajektória je špecifická cesta pohybu mobilnej stanice počas simulácie [3]. V OM môže byť definovaná ako segmentovo orientovaná, alebo vektorovo orientovaná.

- Segmentovo orientovaná trajektória definuje pohyb použitím série preddefinovaných bodov
- Vektorovo orientovaná trajektória definuje pohyb v zmysle chovania, traťovej rýchlosti a intenzity stúpania.

3.2.1 Segmentovo orientovaná trajektória

Segmentovo orientovaná trajektória pozostáva z jedného alebo viacerých hodnôt času presunu medzi bodmi a zo súboru priestorových súradníc, ktoré definujú cestu mobilnej stanice [3]. Trajektórie s premenlivou dĺžkou segmentov obsahujú naviac súbor uhlov, ktoré definujú orientáciu mobilnej stanice v priestore. Segmentovo orientovaná trajektória je uložená v ASCII (American Standard Code for Information Interchange) textovom súbore s koncovkou „.trj“ a je priradená mobilnej stanici alebo podsieti používajúcej atribút trajektórie. Mobilná stanica počas simulácie sleduje jej trajektóriu pohybom po ceste z jedného definovaného bodu na ďalší definovaný bod. Pozícia stanice je v danom čase zistená interpoláciou medzi bodmi trajektórie ihneď pred a po danom čase. Segmentovo orientovaná trajektória špecifikuje umiestnenie mobilnej stanice pre konečnú dobu trvania. Ak simulácia pokračuje mimo posledný špecifikovaný čas v trajektórii, stanica zostane

v poslednom špecifikovanom bode trajektórie. Segmentovo orientovaná trajektória môže byť s pevným časovým intervalom, alebo s premenlivým časovým intervalom.

- V trajektórii s pevným časovým intervalom určuje jedna hodnota časový interval medzi presunmi stanice pre všetky segmenty. Stanici trvá presun medzi každým segmentom rovnaký čas bez ohľadu na dĺžku segmentu. Jedna hodnota obvykle určuje aj výšku pre všetky body.
- V trajektórii s premenlivým časovým intervalom má každý bod špecifikovanú svoju výšku, čas čakania, čas presunu z jedného bodu do druhého a orientáciu v priestore. Čas čakania definuje pauzu stanice v každom bode predtým, ako sa začne pohybovať k ďalšiemu bodu.

3.2.2 Formát súboru segmentovo orientovanej trajektórie

Obidva typy segmentovo orientovanej trajektórie sú založené na ASCII formáte súboru trajektórie pre špecifikovanie trajektórie, vrátane súradníc a časových intervalov presunu [3]. Vytvorenie súboru s trajektóriou je možné v nejakom textovom editore, alebo graficky pomocou „Project Editor“. Formát trajektórie pre premenlivý časový interval a pevný časový interval je nasledovný:

- Formát súboru trajektórie pevného časového intervalu

```
<coordinate_count>
locale: <locale>
<sample_time_step>
<position_unit>
<coordinate_method>
<x_coord_0>, <y_coord_0>, <alt_0>
<x_coord_1>, <y_coord_1>, <alt_1>
…
<x_coord_n>, <y_coord_n>, <alt_n>
```

- Formát súboru trajektórie premenlivého časového intervalu

```
Version: 3
Position_Unit: <position_unit>
Altitude_Unit: <altitude_unit>
Coordinate_Method: <coordinate_method>
Altitude_Method: <altitude_method>
locale: <locale>
Calendar_Start: <start_time>
Coordinate_Count: <coordinate_count>
X Position, Y Position, Altitude, Traverse Time, Wait Time, Pitch,
Yaw ,Roll
<x_coord_0>, <y_coord_0>, <alt_0>, <trav_time_0>, <wait_time_0>,
<pitch_0>,  <yaw_0>, <roll_0)>
<x_coord_1>, <y_coord_1>, <alt_1>, <trav_time_1>, <wait_time_1>,
<pitch_1>,  <yaw_1>, <roll_1)>
…
<x_coord_n>, <y_coord_n>, <alt_n>, <trav_time_n>, <wait_time_n>,
<pitch_n>,  <yaw_n>, <roll_n)>
```

Jednotlivé položky, ktoré sa nachádzajú v súbore s trajektóriou, sú popísané v kapitole 3.2.3

3.2.3 Premenné súboru trajektórie

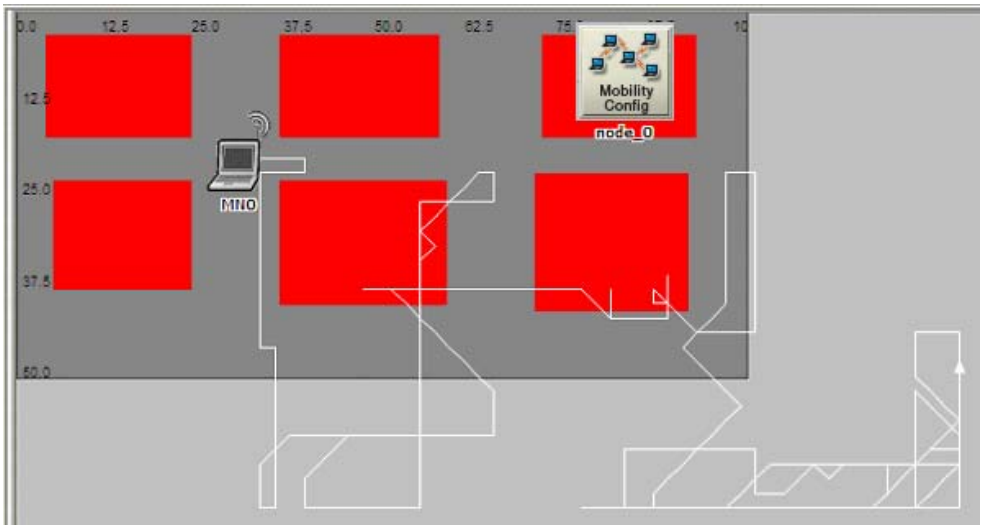
V tab. 3.1 sú uvedené premenné súboru trajektórie. Premenné sú uvedené v stĺpci pod názvom poľa. Ich možné hodnoty sú uvedené v stĺpci popis. Obr. 3.4 a 3.5 popisujú parameter „coordinate_method“.

Tab. 3.1: Premenné súboru trajektórie

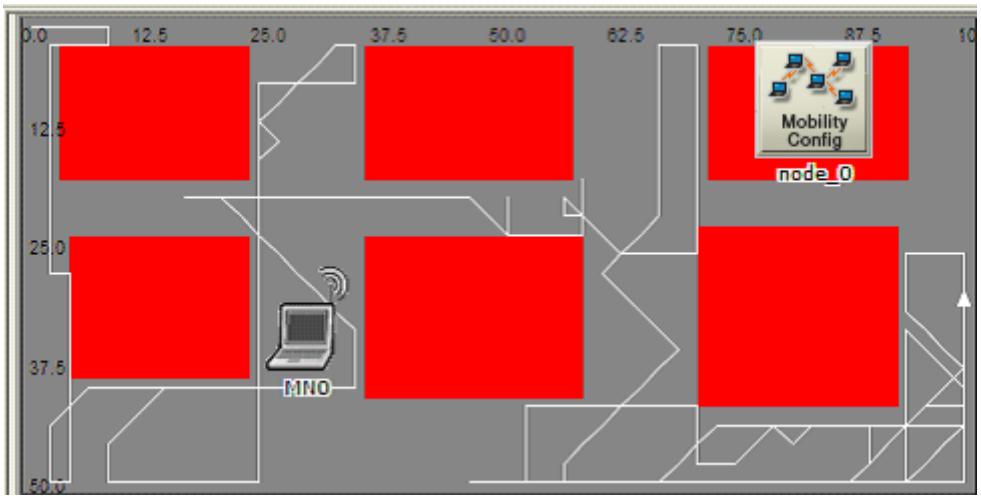
Názov pola	Kľúčové slovo	Popis
altitude_method	Altitude_Method	Zastaralé. Nastavenia podobne ako v „coordinate_method“.
altitude_unit	Altitude_Unit	Jednotka pre alt_n. Platné hodnoty sú „Kilometers“, „Meters“, „Miles“, „Feet“ a „Local“. V prípade „Local“ trajektória použije súradnicový systém podsiete, v ktorej sa stanica nachádza.
alt_n		Absolútna výška nad morom pre pozíciu n. Je typu double.
calendar_start	Calendar_Start	Rezervované pre budúce použitie. Jediná platná hodnota je „unused“.
coordinate_count	Coordinate_Count	Počet pozícií definovaných v súbore trajektórie. Je typu integer.
coordinate_method	Coordinate_Method	Špecifikuje ako sú interpretované hodnoty x_coord a y_coord pozícií. Platné hodnoty sú: • „relative“ – x a y súradnice sú interpretované ako ofset od počiatočnej pozície stanice relatívne k jej rodičovskej podsieti. Prvá pozícia v súbore musí byť 0,0. • „fixed“ – x a y súradnice sú interpretované ako absolútne súradnice vnútri rodičovskej podsiete. Na obr. 3.4 a 3.5 sú znázornené obidve metódy interpretácie súradníc.
locale	locale	Rezervované pre budúce použitie. Jediná platná hodnota je „C“
pitch_n		Čelný náklon v priestore pre pozíciu n. Platné hodnoty sú: • „Degrees“ – hodnota 0 znamená náklon paralelne k zemi. Kladné hodnoty reprezentujú náklon špičkou od zeme, záporné hodnoty reprezentujú náklon špičkou do zeme. • „autocomputed“ – hodnoty náklonu sa nastavujú tak, že sa orientácia stanice zhoduje s vektorom pohybu pre každý segment trajektórie. • „unspecified“ – rovnaké ako hodnota 0.
position_unit	Position_Unit	Jednotky pre x_coord_n a y_coord_n hodnoty

		pozície n. Platné hodnoty sú „Degrees“, „Kilometers“, „Meters“, „Miles“, „Feet“ a „Local“. V prípade „Local“ trajektória použije súradnicový systém podsiete, v ktorej sa stanica nachádza. Podsieť stanice, v ktorej sa stanica nachádza, ovplyvňuje pohyb stanice iba v tom prípade, ak je trajektória podsiete definovaná v nestupňových jednotkách. Viac v relatívnom pohybe.
roll_n		Bočný náklon v priestore pre pozíciu n. Platné hodnoty sú: • „Degrees“ – hodnota 0 znamená náklon paralelne k zemi. Kladné hodnoty reprezentujú náklon pravou stranou vzhľadom k zemi, záporné hodnoty reprezentujú náklon ľavou stranou vzhľadom k zemi. • „unspecified“ – rovnaké ako hodnota 0.
sample_time_step		Čas presunu medzi segmentmi pre formát trajektórie s pevným časovým intervalom
trav_time_n		Čas presunu z pozície n-1 na pozíciu n vyjadrený v štandardnom časovom formáte (6:18:59), alebo v HMS formáte (6h18m59s). HMS formát nevyžaduje všetky tri polia: 6h:18m je platná hodnota. Desatinné čísla sú povolené iba v krajnej pravej jednotke: 6:18:59.5 a 18.5m sú platné hodnoty, ale 6.5:18:59 a 6h18.5m59s platné nie sú. Ak sú zadané iba jednoduché hodnoty ako 23 alebo 59, hodnoty sú interpretované ako sekundy.
version_number	Version	Verzia formátu súboru. Je typu double. Súbor verzie 3 má kladnú os Y od nuly dole, verzia 4 má kladnú os Y od nuly hore. Podľa verzie OM treba prispôsobiť verziu vytváraného súboru s trajektóriou. Pre správnu funkčnosť v súčastnej verzii 16 OM musí byť nastavené na hodnotu „4“, nakoľko je kladná osa Y smerom od nuly hore. V starších verziách OM je os Y naopak.
wait_time_n		Časový úsek, po ktorý stanica ostane na danej pozícii n, než pokračuje na ďalšiu pozíciu. Rovnaký formát ako v „trav_time_n“.
x_coord_n		Súradnica X pre pozíciu n. Je typu double. Ak sú jednotky v „Degrees“, vyjadruje zemepisnú dĺžku.
y_coord_n		Súradnica Y pre pozíciu n. Je typu double. Ak sú jednotky v „Degrees“, vyjadruje zemepisnú šírku.
yaw_n		Natočenie pre pozíciu n. Platné hodnoty sú: • „Degrees“ – hodnota 0 znamená

		natočenie na sever. Kladné hodnoty reprezentujú natočenie na východ, záporné hodnoty reprezentujú natočenie na západ. • „autocomputed“ – hodnoty natočenia sa nastavujú tak, že sa orientácia stanice zhoduje s vektorom pohybu pre každý segment trajektórie. • „unspecified“ – rovnaké ako hodnota 0.



Obr. 3.4: Interpretácia „Coordinate_Method: relative“



Obr. 3.5: Interpretácia „Coordinate_Method: fixed“

3.2.4 Relatívny pohyb

Mobilné stanice môžu byť v OM vložené do podsietí [3]. Z tohto dôvodu môže pohyb podsiete ovplyvňovať pohyb staníc v podsieti. Tento fakt musí byť zvážený pri vytváraní segmentovo orientovaných trajektórií. Efekt pri pohybe je nasledovný:

- Ak sú jednotky pozície v stupňoch, pohyb sa v absolútnom zmysle považuje za relatívny k zemi. Pohyb podsiete nemá žiadny účinok na pohyb staníc v nej.

- Ak jednotky pozície nie sú v stupňoch, napríklad v metroch alebo stopách, pohyb je relatívny k podsieti, v ktorej sa stanica nachádza. Ak sa podsieť pohybuje 10 metrov za sekundu na východ a stanica v nej 5 metrov za sekundu na západ, výsledný pohyb stanice vzhľadom k zemi je 5 metrov za sekundu na východ.

3.2.5 Definovanie trajektórie cez Projekt editor

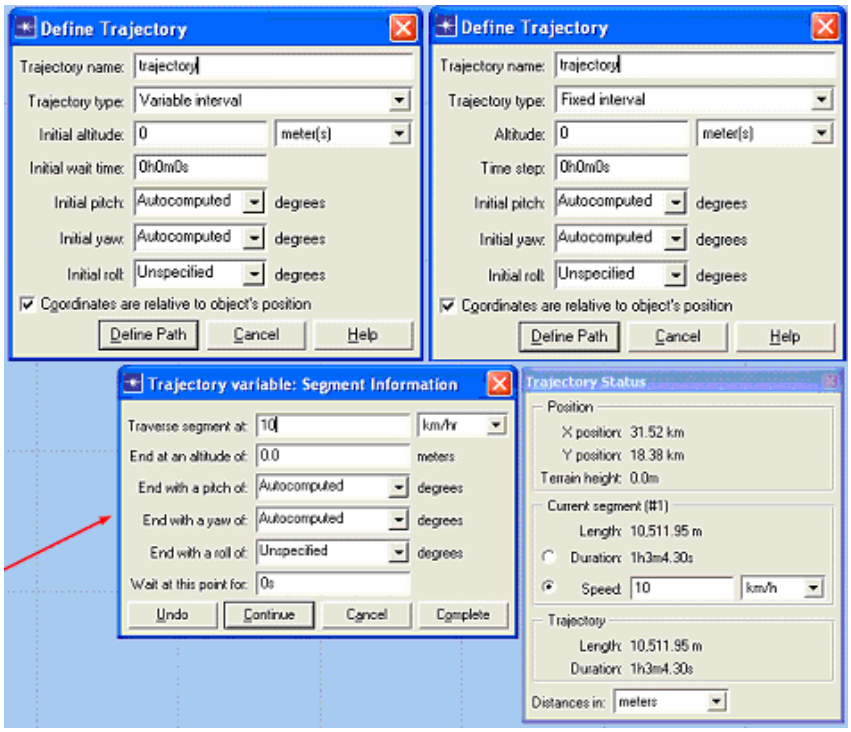
Okrem manuálneho vytvárania trajektórie je možné v OM definovať segmentovo orientované trajektórie graficky pomocou „Project Editor“ [3]. Výsledná trajektória je uložená ako ASCII „*.trj“ súbor, ktorý je možné priradiť mobilnej stanici. Postup pre definovanie trajektórie je trochu odlišný v závislosti na tom, či je trajektória s pevným časovým intervalom, alebo s premenlivým časovým intervalom. Postup pre vytvorenie je nasledovný:

1. V „Project Editor“ vyberieme „Topology > Define Trajectory“
2. Do „Trajectory name“ zadáme názov trajektórie.
3. Z „Trajectory type“ vyberieme typ trajektórie, ktorý chceme definovať. Ďalšie polia sa prispôsobia nášmu výberu.
4. Do kolónky „Altitude“ alebo „Initial altitude“ zadáme výšku prvej pozície a vyberieme jednotky výšky z roletového menu.
5. Pre premenlivý časový interval zadáme počiatočný čas čakania počiatočnej pozície do kolónky „Initial wait time“. Čas čakania aktuálnej pozície spôsobí čakanie stanice na aktuálnej pozícii po stanovený čas, než sa začne pohybovať k ďalšiemu bodu. Syntax hodnôt je obdobná ako v premennej „trav_time_n“ popísanej v časti premenné súboru trajektórie.
6. Pre pevný časový interval zadáme čas presunu medzi segmentami do kolónky „Time step“. Čas presunu definuje čas, za ktorý sa stanica presunie medzi každým segmentom trajektórie.
7. Nastavíme hodnoty „pitch“, „yaw“ a „roll“ prvého bodu. Premenné sú vysvetlené v kapitole 3.2.3.
8. Podľa interpretácie X a Y súradníc zatrhneme pole „Coordinates are relative to object’s position“. Viac viď 3.2.4 a popis „coordinate_method“ v tab. 3.1.
9. Pokračujeme kliknutím na tlačidlo „Define Path“. Dialógové okno statusu trajektórie otvorí tenkú priečnu čiaru , ktorá sa objaví nad kurzorom v pracovnom priestore projektového editoru.
10. Definujeme body trajektórie nasledujúcim spôsobom:

    - Pevný časový interval – ľavým klikom v želanom umiestnení označíme všetky body trajektórie. Po definovaní všetkých bodov klikneme pravým tlačidlom

myši do pracovného priestoru a vyberieme „Complete trajectory definition“ pre ukončenie definície trajektórie.

- Premenný časový interval – ľavým klikom v želanom umiestnení označíme prvý bod trajektórie. Objaví sa informačné okno segmentu. Do neho zadáme hodnoty času presunu alebo rýchlosti z predchádzajúceho bodu do aktuálneho bodu. Ďalej zadáme konečné hodnoty v bode pre premenné „altitude“, „pitch“, „yaw“, „roll“ a „wait time“. Čas presunu medzi bodmi je možné nahradiť rýchlosťou stanice, ktorá sa nastavuje v roletovom menu premennej. Pri zadávaní výšky „altitude“ treba použiť rovnaké jednotky ako v prvej hodnote výšky „initial altitude“. Pri zadávaní hodnôt „traversal time“ a „wait time“ použijeme formát popísaný v tab. 3.1. Rozhrania definície bodov sú zobrazené na obr. 3.6.



Obr. 3.6: Rozhrania definície bodov trajektórie

Pre definovanie ďalších bodov klikneme na tlačidlo „Continue“ a špecifikujeme ďalšie body rovnakou cestou. Po definovaní všetkých bodov v trajektórii ukončíme nastavovanie kliknutím na tlačidlo „Complete“.

11. Po ukončení definície trajektórie sa trajektória uloží v ASCII súbore s koncovkou „.trj“ do primárneho adresára modelu.

3.2.6 Nastavenie trajektórie stanici z vytvoreného súbotu *.trj

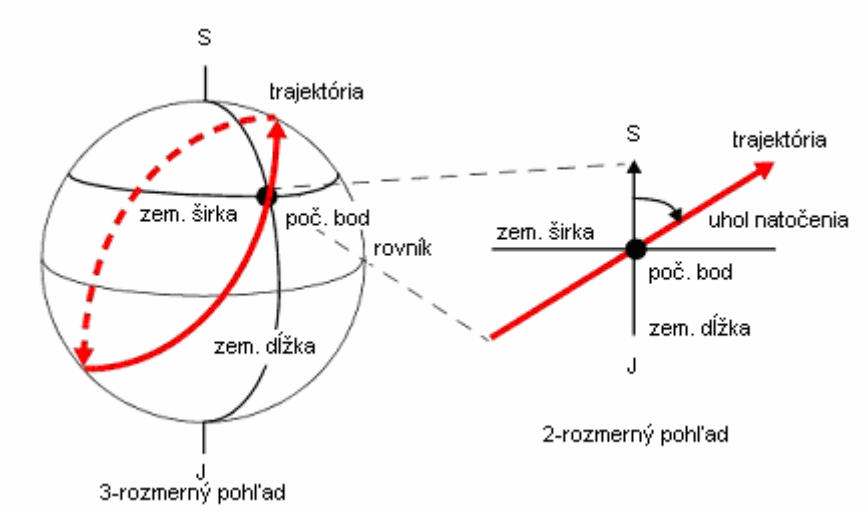
Procedúra priradenia trajektórie jednej alebo viacerým staniciam je nasledovná:

1. Vyberieme mobilné stanice, na ktoré chceme aplikovať náhodné trajektórie.

2. Zvolíme „Topology > Random Mobility > Set Trajectory Created From Random Mobility...". OM vyhľadá pre každú označenú stanicu náhodné trajektórie z aktuálneho scenára a ak nájde, aplikuje ich na vybrané stanice.

3.2.7 Vektorovo orientovaná trajektória

Vektorovo orientované trajektórie [3] pozostávajú zo smeru a rýchlosti, ktorá sa môže meniť v závislosti na čase. Nastavením atribútu trajektórie stanice na „VECTOR" sa špecifikuje použitie vektorovo orientovanej trajektórie. Vektorovo orientovaná trajektória je založená na veľkej kruhovej dráhe okolo Zeme. Pre definovanie cesty sa špecifikuje uhol natočenia kruhovej dráhy „bearing", rýchlosť voči zemi „ground speed" a rýchlosť stúpania „ascent rate". Kruhová dráha je sústredená do stredu Zeme a prechádza cez špecifikované body. Obvykle sa jedná o satelitnú stanicu, ktorá sleduje svoju dráhu. Počas simulácie sa mení zemepisná šírka a dĺžka podľa kruhovej trajektórie. Ako príklad slúži obr. 3.7, na ktorom je zobrazená možná trajektória založená na veľkej kruhovej dráhe okolo Zeme. Zemepisná šírka a dĺžka stanice bude na začiatku simulácie v počiatočnom bode. Uhol natočenia, rýchlosť a stúpanie určujú cestu počas simulácie. Súradnice Zemepisnej šírky a dĺžky sledujú cestu hlavnej kružnice založenej na traťovej rýchlosti stanice a výške, ktorá sa mení v závislosti na rýchlosti stúpania.



Obr. 3.7: Vektorovo orientovaná trajektória

Uhol natočenia „bearing" je použitý iba pre zistenie hlavnej kružnice pretínajúcej počiatočný bod, ktorá bude použitá v trajektórii. Magnetický kurz objektu vyplýva z trajektórie a bude sa meniť v každom bode pozdĺž trajektórie, pričom vektor trajektórie môže zostať rovnaký počas simulácie. Cesta stanice sa môže počas simulácie zmeniť, ak sa zmení uhol natočenia. V tomto prípade vznikne nový počiatočný bod a nová kruhová dráha sa prepočíta z nového počiatočného bodu a uhlu natočenia.

3.3 Priama manipulácia pozície stanice

Ak má stanica špecifikovanú trajektóriu, jej cesta je vopred daná pre celú simuláciu [3]. Ak však nie je špecifikovaná žiadna trajektória, pozícia stanice môže byť priamo aktualizovaná

nejakým procesom v priebehu simulácie. Atribúty X a Y súradníc špecifikujú pozíciu stanice v jej podsieti. Výška stanice určuje relatívny zdvih k morskej hladine. Zmena týchto atribútov spôsobí okamžitú zmenu umiestnenia mobilnej stanice. Pre dynamickú zmenu polohy stanice existujú dve techniky. V obidvoch prípadoch je za modifikovanie atribútu pozície zodpovedný užívateľom definovaný proces. Prvá technika je centralizovaný prístup, v ktorom je jeden proces zodpovedný za aktualizáciu pozície všetkých mobilných staníc v sieťovom modeli. Tento proces je často zabudovaný vo vnútri stanice, ktorá je špeciálne označená ako stanica s centrálnym riadením. Druhý prístup je decentralizovaný prístup, kde má každá stanica svoj vlastný proces. Tento proces aktualizuje iba pozíciu stanice, v ktorej sa nachádza. Príklad vzorového procesu používajúceho decentralizovaný prístup k atribútom pozície je nasledovný:

```
/* Získanie ID objektu rodičovskej stanice. */
parent_node_objid = op_topo_parent (op_id_self ());

/* Získanie nových relatívnych súradníc podsiete pre rodičovskú stanicu.*/
node_position_compute (parent_node_objid, &x_pos, &y_pos, &alt);

/* Nastavenie atribútu pozície rodičovskej stanice na nové hodnoty. */
op_ima_obj_attr_set_dbl (parent_node_objid, "x position", x_pos);
op_ima_obj_attr_set_dbl (parent_node_objid, "y position", y_pos);
op_ima_obj_attr_set_dbl (parent_node_objid, "altitude", alt);
```

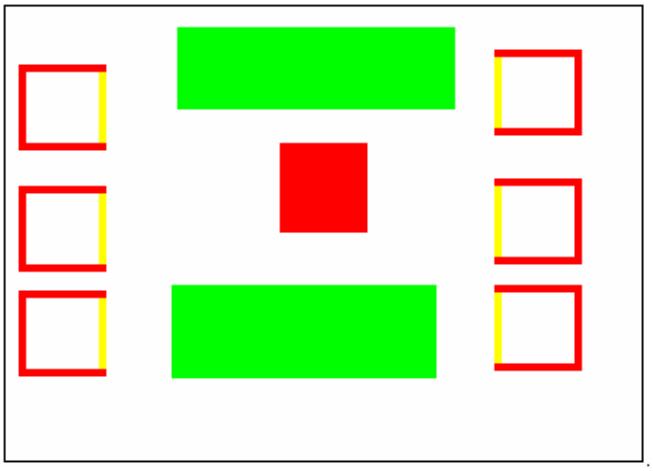
Príklad vzorového procesu používajúceho centralizovaný prístup k atribútom pozície je nasledovný:

```
/* Zistenie počtu mobilných staníc v systéme. */
num_nodes = op_topo_object_count (OPC_OBJTYPE_NDMOB);

/* Pre každú mobilnú stanicu sa získa jej ID objektu, systémové ID */
/* a nová pozícia. Nastavenie atribútov pozície staníc na nové hodnoty. */
for (i = 0; i < num_nodes; i++)
{
/* Získanie ID objektu i-tej mobilnej stanice. */
node_objid = op_topo_object (OPC_OBJTYPE_NDMOB, i);

/* Nastavenie atribútu pozície rodičovskej stanice na nové hodnoty. */
op_ima_obj_attr_set_dbl (node_objid, "x position", x_pos);
op_ima_obj_attr_set_dbl (node_objid, "y position", y_pos);
op_ima_obj_attr_set_dbl (node_objid, "altitude", alt);
}
```

Na rozdiel od trajektórií, kde simulačné jadro modeluje súvislý pohyb, priama manipulácia atribútu pozície stanice vykonáva diskrétne zmeny pohybu. Nejedná sa teda o súvislý pohyb, čo môže zapríčiniť nečakané správanie vzhľadom na štandardné rádiové vysielanie a prijímanie stanice. Môžu vzniknúť problémy týkajúce sa zvyšujúceho sa blokovania signálu, interferencie paketových výpočtov a nepresných štatistík kanálu rádiového vysielača. Tieto problémy odkazujú na fakt, keď sú poslané dva pakety z jedného kanálu rádiového vysielača, kde je čas prijatia prvého paketu rovnaký ako čas vysielania druhého paketu. Problémy sú založené na dvoch hodnotách výpočtu oneskorení vplyvom šírenia. Prvá hodnota je oneskorenie medzi odoslaním a prijatím prvého paketu, druhá hodnota je oneskorenie výsielania druhého paketu. Ak nie sú tieto hodnoty oneskorení rovnaké, budú sa pakety nekorektne prekrývať, alebo bude medzi nimi rozdiel. Ak sa dve stanice pohnú smerom k sebe, alebo od seba počas simulácie, bude sa adekvátne meniť oneskorenie vplyvom šírenia. Riešením je výmena štandardného vysielacieho zariadenia iným, ktorý bude korektne počítať oneskorenie vplyvom šírenia na konci prenosu.

4 Mapový podklad v MATLABE

Značná časť práce je venovaná spracovaniu mapového podkladu do takej podoby, aby bola použiteľná pre riadenie pohybu stanice v prostredí OM. Bola vytvorená funkcia v nástroji MATLAB, ktorá spracuje bitmapový obrázok ako mapový podklad. Vzorová podoba vstupného obrázku je uvedená na obr. 4.1.

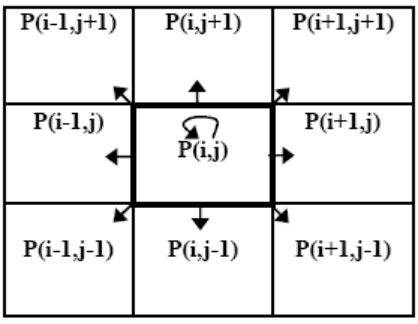


Obr. 4.1: Vstupný obrázok

Funkcia rozoznáva 4 farby, ktorých pravdepodobnosti vstupu sú vopred nadefinované. Na obrázku sú červenou farbou znázornené myslené steny budov, ktoré sú nepriechodné. Žltá farba reprezentuje vchody do budov s nastavenou pravdepodobnosťou vstupu. Zelené plochy reprezentujú trávnaté plochy alebo parky s malou pravdepodobnosťou vstupu. Plocha s bielou farbou reprezentuje myslenú komunikáciu, ktorá ma plný prístup. Na základe spracovaného vstupného obrázka a vstupných premenných funkcie sa vygeneruje trajektória, ktorá sa uloží do výstupného súboru s vhodnou hlavičkou. Vstupy a výstupy funkcie sú popísané v kapitole 4.5. Súbor sa uloží pod názvom „*trajectory_variable.trj*“ do definovanej zložky, v tomto prípade do „C:/“.

4.1 Popis pohybu po mape

Pohyb stanice je vo vytvorenej funkcii MATLABu implementovaný pomocou Markovej mriežky, viď obr. 4.2 [6].



Obr. 4.2: Markova mriežka

Každé pole má svoju pravdepodobnosť výskytu nazývanú váha. V prvom kroku sa určí smer na základe pravdepodobností políčok. Následne sa vykoná pohyb v určenom smere a nastaví sa aktuálna mriežka na novú. Nová mriežka bude mať prispôsobené váhy políčok tak, aby bol v prípade voľného pohybu uprednostňovaný pohyb v smere predchádzajúceho pohybu. V prípade použitia vyhľadávania hotspotov bude uprednostnený pohyb v smere k vyhľadávanému hotspotu. Pre každý z deväť smerov je zadefinovaná vlastná mriežka s váhami. Príklad definície matice s nastavenými váhami pre smer doprava je na obr. 4.3.

```
mat_R  = [ 0 , 0.03 , 0.03 ;
           0 , 0.01 , 0.79 ;
           0 , 0.03 , 0.03 ];
```

Obr. 4.3: Definícia matice smeru doprava

Z pravdepodobností políčok je vidieť preferovaný smer doprava. Cyklus pohybu sa opakuje, pokiaľ sa nevykoná definovaný počet krokov. V prípade použitia vyhľadávania hotspotov sa cyklus ukončí po dosiahnutí posledného nastaveného hotspotu.

4.2 Popis vlastnej funkcie pre spracovanie bitmapy

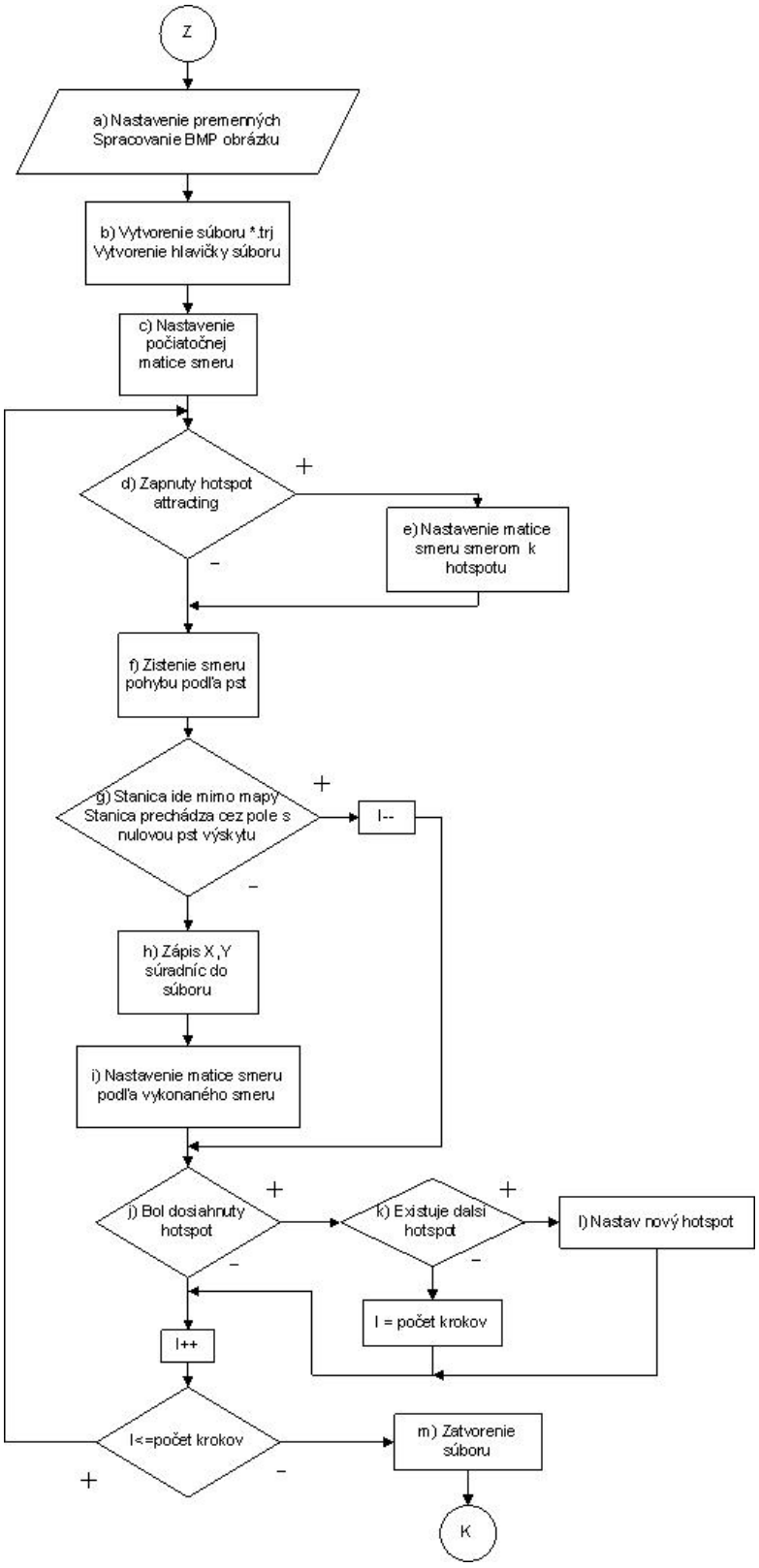
Následujúce riadky zobrazujú, ako je v MATLABe riešené spracovanie bitmapového obrázka do modelu RGB (Red Green Blue), vhodného pre ďalšie spracovanie [4, 5]:

```
function[matica] = nacitanie_obr(krok);
% načítanie obrázku
[A] = imread('obr2.bmp');
size(A);
%zobrazenie obrázku
figure(1);
image(A);
% nastavenie pomeru strán 1:1
daspect([1 1 1]);
hold on;
% vytvorenie nulovej matice o veľkosti obrázka
velkost_obr = size(A);
matica = zeros(velkost_obr(1)+2*krok,velkost_obr(2)+2*krok,1);
% nastavenie pst pre každý bod obrázku
for i = 1:1:velkost_obr(1)
    for j = 1:1:velkost_obr(2)
        % ak je bod červeny (255 0 0), nastaví pst 0%
        if A(i,j,1) == 255 && A(i,j,2) == 0 && A(i,j,3) == 0;
           matica(i+krok,j+krok,1)= 0.0;
        % ak je bod biely (255 255 255), nastaví pst 100%
        elseif A(i,j,1) == 255 && A(i,j,2) == 255 && A(i,j,3) == 255;
           matica(i+krok,j+krok,1)= 1.0;
        end
    end
end
```

Spôsob spracovania je riešený vlastnou funkciou, ktorej vstupom je obrázok a výstupom trojrozmerné pole s priradenými hodnotami pravdepodobností ku každej súradnici bodu. Toto pole je následne násobené smerovými maticami Markovej mriežky.

4.3 Popis funkcie pre generovanie súboru *.trj

Na obr. 4.4 je zobrazený vývojový diagram funkcie v MATABe pre generovanie súboru s trajektóriou.



Obr. 4.4: Vývojový diagram funkcie pre generovanie súboru s trajektóriou

4.4 Popis vývojového diagramu

V nasledujúcej kapitole sú popísané jednotlivé bloky vývojového diagramu. Popis bloku:

a) V prvom bloku sa nastavia nasledovné vstupné premenné:

- *počet vykonaných krokov* **i** – premenná, v ktorej je uložený vykonaný počet krokov. Počiatočná hodnota je nastavená na 1.

- *matica smeru pohybu* **smer** – do matice smeru sa priebežne ukladajú hodnoty 1 – 9 reprezentujúce matice s nastavenými smermi pohybu. Premenná smer je odkazom na dané matice s vlastnými hodnotami pravdepodobností. Matíc je deväť, pre každý smer pohybu jedna, vrátane matice státia.

- *súradnice počiatočného bodu* **x_coord** a **y_coord** – premenné počiatočného bodu reprezentujú bod, z ktorého bude pohyb po mape vychádzať.

- *počet krokov* **pocet_krokov** – v premennej je nastavený počet krokov, ktorý sa má vykonať.

- *veľkosť kroku* **krok** – hodnota, o ktorú sa objekt presunie pri pohybe z bodu A do bodu B

- *smerové matice* **mat_X** – matice s vlastnými hodnotami pravdepodobností. Podľa pravdepodobností sa vygeneruje nasledujúci smer pohybu z daného miesta. Namiesto písmena X má každá matica svoje označenie pre svoj smer.

  Premenné potrebné pre funkciu hotspotov:

- *súradnice hotspotov* **x0_coord a y0_coord** – vektory, v ktorých sú uložené súradnice hotspotov.

- *index hotspotu* **hotspot_index** – index aktuálneho hotspotu, ku ktorému je stanica priťahovaná.

- *počet hotspotov* **pocet_hotspotov** – vyjadruje počet celkových hotspotov, ktoré sa na mape nachádzajú.

- *polomer kružnice* **R** – hodnota polomeru kružnice, ktorá vyjadruje oblasť dosiahnutia hotspotu

- *atribút zapnutia hotspotov* **Hotspot_attracting** – tag zapnutia hotspotov, ak je hodnota 0, generuje sa náhodný pohyb. Ak je 1, generuje sa trasa k najbližšiemu nastavenému hotspotu.

- *rozdielové hodnoty* **DIF_X a DIF_Y** – premenné, v ktorých je uložený rozdiel hodnôt aktuálnych X a Y súradníc a súradníc aktuálneho nastaveného hotspotu. Hodnoty sú použité pri výpočte tolerancie dosiahnutia hotspotu.

Po nastavení premenných skriptu sa ako prvá zavolá vytvorená pomocná funkcia pre spracovanie obrázka do vhodnej podoby pre ďalšie spracovanie „*nacitanie_obr(krok)"*.

Táto funkcia je popísaná v nasledujúcom texte. Vstupný obrázok vo formáte „*.bmp" sa načíta do premennej **A** vo forme matice pomocou funkcie „*imread"*. Matica **A** je trojrozmerná. Prvá a druhá hodnota matice predstavuje súradnice bodov obrázka. Jedná sa o jednorozmerné polia. K týmto súradniciam je pridružené tretie trojrozmerné pole, ktoré reprezentuje RGB hodnoty bodu súradnice obrázka. RGB hodnoty sú pre každú zložku v rozmedzí 0 – 255 pre daný formát bitmapového obrázka. Ďalej sa nastaví pomer strán obrázka na 1:1. Do premennej **velkost_obr** sa uloží veľkosť spracovávaného obrázka. Vytvorí sa nulová matica **matica** o veľkosti obrázka zväčšeného o hodnotu **krok**. Matica je zväčšená z toho dôvodu, aby bolo možné neskoršie násobenie matíc na okrajových polohách obrázka. Pomocná matica **matica** je trojrozmerná. Prvé dve hodnoty reprezentujú súradnice bodov, pričom tretia hodnota reprezentuje neskôr nastavenú pravdepodobnosť vstupu na dané pole. V cykle sa prejde postupne každé pole, pričom sa testuje farba bodu podľa jeho RGB hodnôt. Je možné nastaviť pravdepodobnosť vstupu na pole v rozmedzí 0 – 100 percent pre farby vyjadriteľné RGB hodnotami. Hodnoty pravdepodobností boli testovaním získané tak, aby jednotlivé farby odpovedali reálnym objektom ako je stena budovy, trávnik, dvere a cesta. Pravdepodobnosti sú nastavené pre farby nasledovne:

Červená	(R=255, G=0, B=0)	pst = 0.0	0	percent	stena
Zelená	(R=0, G=255, B=0)	pst = 0.1	10	percent	trávnik
Žltá	(R=255, G=255, B=0)	pst = 0.9	90	percent	dvere
Biela	(R=255, G=255, B=255)	pst = 1.0	100	percent	cesta

Výstupom funkcie „*nacitanie_obr(krok)"* je matica **matica**, v ktorej sú uložené pravdepodobnosti vstupu každého poľa.

b) Pomocou funkcie „*fopen"* sa vytvorí súbor vo formáte „*.trj", do ktorého sa budú následne zapisovať hodnoty trajektórie vo vhodnom formáte. Formát súboru „*.trj" je popísaný v kapitole 3.2.3. Vytvorí sa hlavička súboru podľa formátu. Dáta sa do súboru zapisujú funkciou „*fprintf*".

c) Do pomocnej premennej **mat_SMERU** sa uloží počiatočná matica s hodnotami pravdepodobností. Ak je povolená funkcia hotspotov, vykreslia sa ich body. Začne sa cyklus, ktorý kontroluje či je počet vykonaných krokov **i** menší, alebo rovný, ako počet nastavených krokov **pocet_krokov**.

d) Testuje sa premenna **Hotspot_attracting**. Ak je rovný 0, preskočí sa priťahovanie k aktuálne nastavenému hotspotu. Ak je rovný 1, pokračuje sledovanie aktuálne nastaveného hotspotu.

e) Zisťuje sa pozícia aktuálnej pozície vzhľadom k hotspotu. Podľa aktuálnej polohy sa nastaví smerová matica pomocou cyklov *„if"* tak, aby jej smer smeroval k hotspotu.

f) Vygeneruje sa náhodné číslo a uloží do premennej **r_smer**. Toto číslo bude potrebné neskôr pri generovaní nového smeru pohybu. Do pomocnej premennej **mat_temp** sa uloží výsek z matice, v ktorej je uložený spracovaný obrázok s pravdepodobnosťami. Výsek sa prevedie podľa polohy počiatočných súradníc. Daná matica sa vynásobí s maticou smeru pohybu. Následne sa nová matica podelí sumou jej hodnôt pravdepodobností, čím sa dostanú absolútne hodnoty pravdepodobností po prenásobení matíc. Operácia je vyjadrená vzťahom

$$mat\_Pom\_abs = \frac{mat\_Pom}{sum(sum(mat\_Pom))}, \tag{4.1}$$

kde *mat_Pom_abs* predstavuje maticu s absolútnymi hodnotami pravdepodobností. Matica *mat_Pom* predstavuje maticu s hodnotami vynásobených matíc **mat_temp** a **mat_SMERU**. Absolútne hodnoty pravdepodobností sa postupne vyčítajú do vektoru.

Pomocou náhodného čísla **r_smer** sa otestuje vektor s hodnotami. Výsledkom je smer pohybu, ktorý sa má vykonať v nasledujúcom kroku.

g) Nasleduje testovanie smeru pohybu pomocou funkcie „*if*“. Pre každý smer je zadefinovaný podobný súbor príkazov pre kontrolu a vykonávanie pohybu. Vykonávanie každého smeru sa líši iba v zápise hodnôt do súboru podľa smeru. Po vyhodnotení smeru, ktorý sa má vykonať, sa začne daný smer pohybu vykonávať. Pred zápisom hodnôt nového bodu do súboru sa otestujú určité podmienky. Zisťuje sa, či sa medzi starou a novou polohou nenachádza pole s nulovou pravdepodobnosťou vstupu. Ďalej sa otestuje, či sa nová poloha, na ktorú sa má v mape vstúpiť, nachádza v mape a nie mimo mapy. Pohyb na nový bod v mape sa nevykoná v prípade, že sa jedna z podmienok nesplní.

h) Po splnení podmienok sa do súboru zapíšu nové súradnice v určitom formáte.

i) Do premennej **mat_SMERU** sa uloží matica odpovedajúca vykonávanému smeru a zvýši sa počet vykonaných krokov **i**.

j) Pomocou rovnice kružnice

$$r^2 = x^2 + y^2 \tag{4.2}$$

sa testujú aktuálne súradnice **x_coord** a **y_coord** so súradnicami hotspotu **x0_coord** a **y0_coord**. Podmienka je splnená, ak sa súradnice hotspotu nachádzajú vnútri kružnice s polomerom **R** a so stredom **x0_coord** a **y0_coord.**

k) Testuje sa existencia ďalšieho hotspotu. Ak ďalší hotspot neexistuje, nastaví sa počet vykonaných krokov **i** na nastavený počet krokov, čoho dôsledkom sa cyklus funkcie ukončí.

l) Nastavia sa súradnice ďalšieho hotspotu.

m) Cyklus končí pred blokom m, ak sa počet vykonaných krokov **i** rovná počtu nastavených krokov **pocet_krokov**. Súbor „*.trj“ sa zavrie pomocou funkcie „*fclose*“.

4.5 Použité funkcie

V tejto časti sú zhrnuté niektoré dôležité funkcie z vytvorenej funkcie v MATLABe pre generovanie súboru s trajektóriou [4, 5]:

- Funkcia pre zistenie veľkosti vstupného obrázka. Podľa veľkosti sa ošetruje zamedzenie pohybu stanice mimo súradníc obrázka, teda mimo mapy:

```
velkost_obr = size(matica);
```

- Funkcia pre vytvorenie nového súboru s názvom „trajectory_variable.trj“. Ak súbor s rovnakým názvom už existuje, otvorí súbor pre zápis a dáta sa prepíšu:

```
fid = fopen('trajectory_variable.trj','+w');
```

- Funkcia pre zápis dát do súboru. V tomto prípade sa jedná o zápis premennej „Position_Units“ s hodnotou „Meters“:

```
fprintf(fid,'Position_Unit: Meters\n');
```

- Funkcia pre vykreslenie čiary z predchádzajúceho bodu na bod aktuálny. Čiara je vykreslená modrou farbou so šírkou 2 podľa premenných „-b“ a „2“:

```
plot(x_coord,y_coord,'-b','LineWidth',2);
```

- Funkcia pre zápis hodnôt do súboru. Vkladané hodnoty sú súradnice stanice X, Y a výška stanice:

```
fprintf(fid,   '%1.15f   ,%1.15f   ,%d   ,0h0m5s   ,0h0m0s   ,0   ,0   ,0
\n',x_coord(i), y_coord(i), alt(i));
```

- Funkcia pre zatvorenie a uvolnenie vytváraného súboru:

```
fclose('all');
```

4.6 Vstupy a výstupy funkcie

Funkcia spracováva vstupný obrázok a generuje trajektóriu podľa kritérií, ktorými sú vstupné premenné funkcie. Funkcia je zadefinovaná nasledovne:

```
function [x_coord,y_coord,pocet] =
zapis_variable_func(Xp,Yp,Pkrok,X0,Y0,Hpoc,Rpol,Hatr,Id);
```

Vstupné premenné funkcie sú:

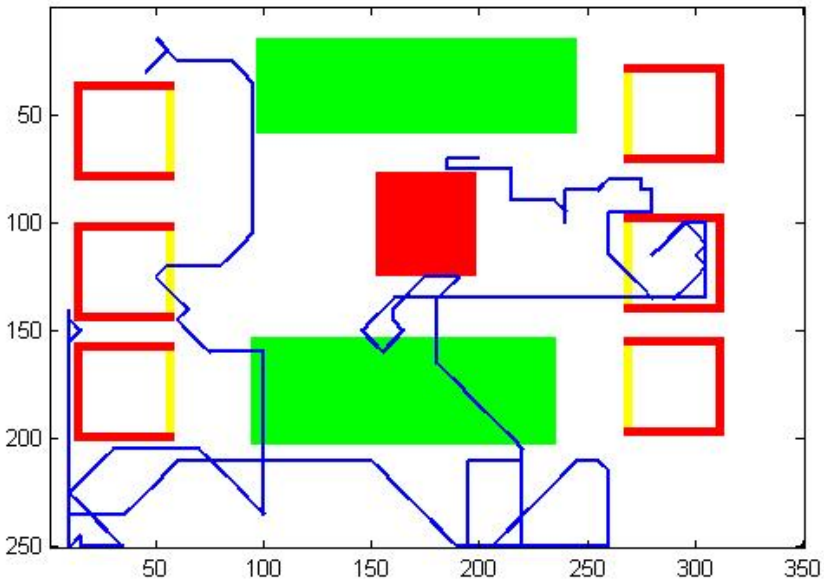
Xp – počiatočná X súradnica
Yp – počiatočná Y súradnica
Pkrok – maximálny počet krokov pre generovanie súradníc
X0 – vektor X súradníc hotspotov
Y0 – vektor Y súradníc hotspotov
Hpoc – počet hotspotov
Rpol – polomer dosiahnutia hotspotu
Hatr – atribút zapnutia vyhľadávania hotspotov
Id – identifikačné číslo stanice, ktoré slúži pre identifikáciu výstupných obrázkov

Výstupné premenné funkcie sú:

x_coord – vektor s hodnotami vygenerovaných X súradníc
y_coord – vektor s hodnotami vygenerovaných Y súradníc
pocet – počet vygenerovaných hodnôt X a Y súradníc

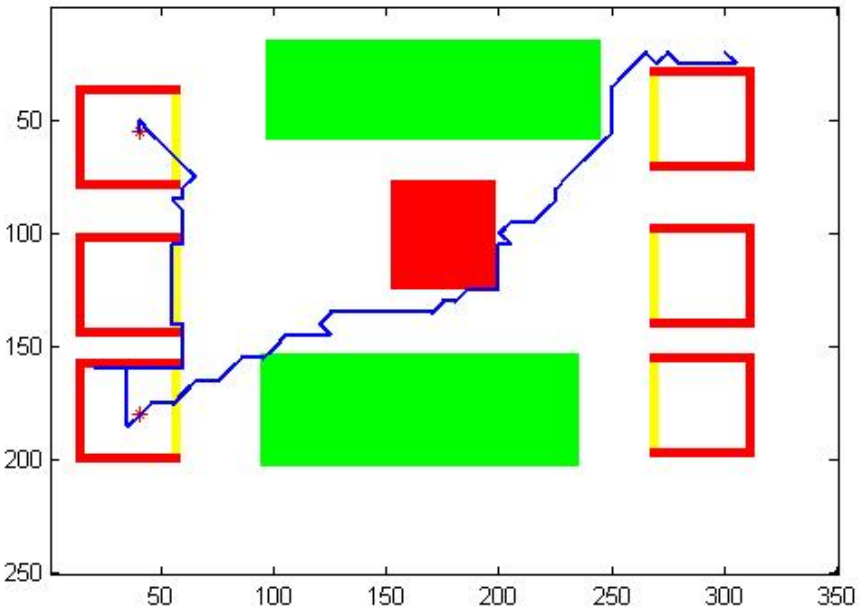
Tieto výstupné premenné majú význam pre priame riadenie pozície stanice, kde sa ich hodnoty predávajú z MATLABu do OM. V prípade riadenia pomocou súborov s trajektóriami sú tieto premenné nepoužité. Ich hodnoty sú zapísané v súbore s trajektóriou.

Výstupom funkcie je súbor s trajektóriou pre ďalšie spracovanie. Pre grafické zobrazenie trajektórie slúži výstupný obrázok, ktorý sa skladá zo vstupného obrázka a vygenerovanej trajektórie. Na obr. 4.5 je zobrazený výstupný obrázok s vygenerovanou náhodnou trasou bez použitia funkcie vyhľadávania hotspotov. Vygenerovaná trasa je znázornená na obrázku a uložená v súbore „trj“.



Obr. 4.5: Výstupný obrázok bez Hotspotov

Na obr. 4.6 je výstupný obrázok pri zapnutej funkcii vyhľadávania hotspotov. Štartovná pozícia sa nachádza vpravo hore. Prvý hotspot sa nachádza vľavo dole. Po jeho dosiahnutí sa nastaví ako aktívny ďalší hotspot, ktorý sa nachádza vľavo hore. Hotspoty sa dosiahnú v prípade, že sa medzi nimi a aktuálnymi súradnicami nenachádza slepá cesta. To sa môže stať v prípade, že stanica vstúpi do budovy a ďalší bod sa nachádza za uzatvorenou stenou z oboch strán.



Obr. 4.6: Výstupný obrázok s nastavenými hotspotmi

Celý zdrojový kód bol vytvorený vo verzii MATLAB 7.5 (R2007b) bez inštalácie pomocných balíkov. Zdrojový kód funkcie pre generovanie trajektórie v MATLABe sa nachádza v prílohe A.

5 Model s použitím súborov trj

V tejto kapitole je popísané vytvorenie modelu s riadením pohybu stanice v simulačnom prostredí OM podľa mapového podkladu. Model sa skladá z troch staníc pre MANET siete. Každá stanica využíva pre riadenie jej pohybu vopred pripravený súbor s trajektóriou, ktorý bol vygenerovaný na základe mapového podkladu spracovaného v MATLABe. Projekt je vytvorený v OM verzia 16. Pre vytvorenie projektu sú potrebné vstupné súbory s trajektóriami a obrázok na pozadie pracovnej plochy vo formáte „tif“. Obrázok je možné uložiť v potrebnom formáte pomocou niektorého z grafických editorov.

5.1 Príprava potrebných súborov

Pre vytvorenie projektu sú potrebné vstupné súbory vygenerované z nástroja MATLAB na základe mapového podkladu. Súbor s trajektóriou sa vygeneruje príkazom, ktorý spustí funkciu s parametrami. Nasledujúce príkazy reprezentujú vygenerovanie trajektórii:

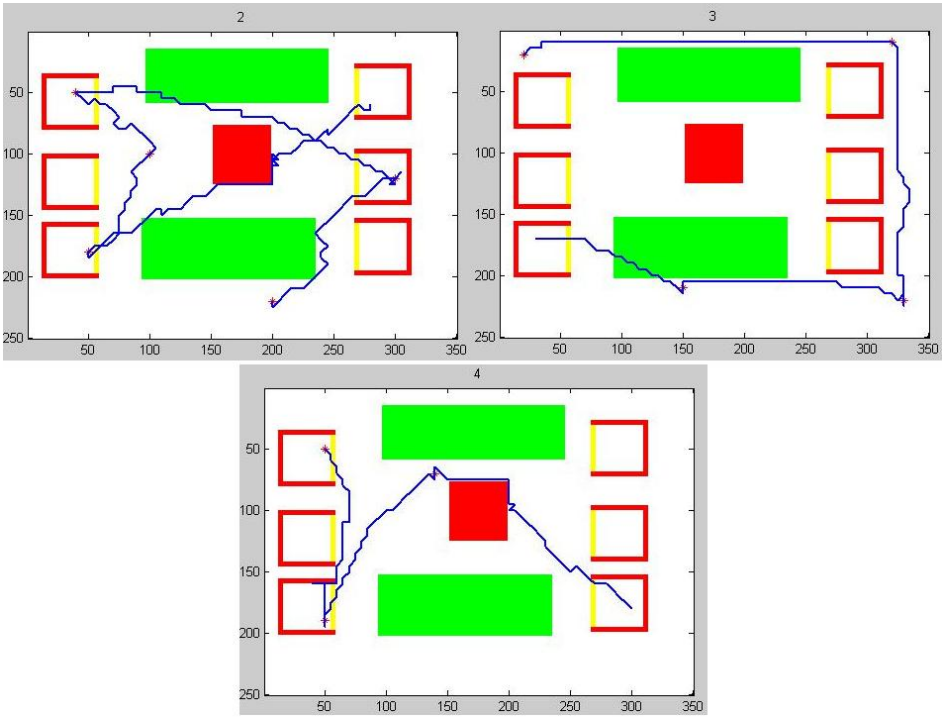
1. súbor:
zapis_variable_func(280,60,400,[50,100,40,300,200], [180,100,50,120,220],5,5,1,2);

2. súbor:
zapis_variable_func(30,170,400,[150,330,320,20], [210,220,10,20],4,5,1,3);

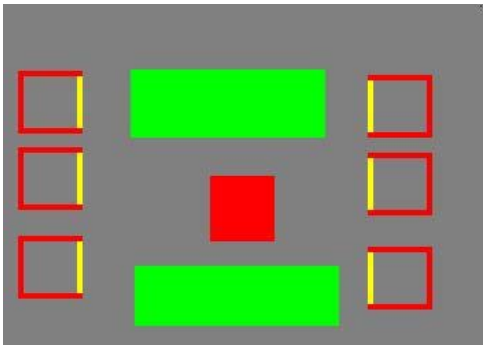
3. súbor:
zapis_variable_func(300,180,400,[140,50,50], [70,190,50],3,5,1,4);

Po spustení sa vygeneruje súbor s trajektóriou v zložke „C:/“. Ten treba pomenovať nejakým dočasným názvom, aby sa pri generovaní ďalšieho súboru neprepísal. Následne je možné vygenerovať ďalší súbor. Obdobným spôsobom je možné vygenerovať potrebné súbory podľa uvedených príkazov. V prípade použitia hotspotov je potrebné upraviť vo vygenerovanom súbore hodnotu „Coordinate_Count“ pomocou nejakého textového editora na hodnotu, ktorá odpovedá počtu súradníc vo vygenerovanom súbore. Problém vzniká pretože sa najskôr vytvorí hlavička súboru, kde je nastavený počet súradníc a až potom sa plní súbor hodnotami podľa počtu súradníc. Bez použitia hotspotov sa vygeneruje daný počet súradníc a problém nevzniká. Pri použití hotspotov sa však dopredu nevie koľko súradníc sa vygeneruje a tak sa zapíše maximálny nastavený počet súradníc. Pri použití riadenia pomocou priamej manipulácie pozície stanice tento problém nevzniká, pretože sa súbor trj nepoužíva. Po spustení funkcie sa okrem vygenerovania súboru zobrazí aj grafické zobrazenie trajektórie na mapovom podklade, ktoré je zobrazené obr. 5.1.

Ďalším potrebným súborom je obrázok pozadia projektu, ktorý však nemá žiadny vplyv na funkčnosť projektu. Je potrebný iba pre porovnanie správnosti trajektórie pohybu stanice podľa vstupného mapového podkladu. Obrázok je vhodné pre správne zobrazenie upraviť. Jedná sa o upravenie farby pozadia napríklad na farbu šedú a to z toho dôvodu, aby bolo vidieť trajektórie staníc, ktoré sú v OM vykreslené farbou bielou. V opačnom prípade by farba pozadia splývala s farbou trajektórií a nebolo by možné trajektóriu rozoznať. Obrázok je tiež potrebné otočiť v ose Y, pretože OM opačne zobrazuje os Y vzhľadom k MATLABu. Záleží však aj od verzie OM. Obrázok bol popísaný v kapitole 4. Upravený obrázok pozadia je vidieť na obr. 5.2.



Obr. 5.1: Grafické zobrazenie trajektórie súboru



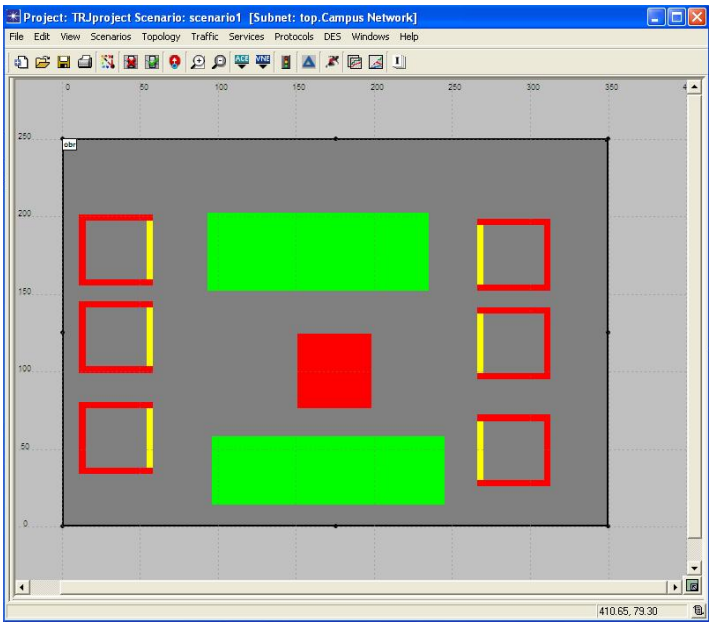
Obr. 5.2: Upravený obrázok na pozadie projektu

5.2 Vytvorenie projektu

Nasledujúci text popisuje vytvorenie projektu:

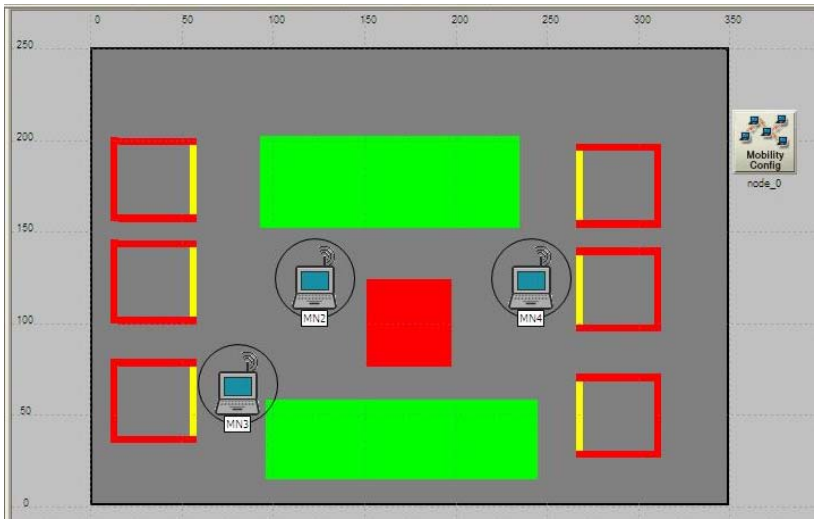
1. Nový projekt v OPNET Modeler sa vytvorí pomocou kontextového menu „File –> new –> project“. Treba vyplniť názov projektu *Project name:* **TRJprojekt** a *scenario*: **scenario1**, ponechať zakliknuté „start up wizard“ a voľbu potvrdiť.
2. Z *initial topology* vyberieme **Create empty scenario** a z *network scale:* **Campus**. Zaklikneme možnosť „use metric units“. Nastavíme veľkosť *Size* na hodnoty *X span*: **350**, *Y span*: **250**, jednotky *units*: **Meters** a voľbu potvrdíme.
3. Z *Model family:* **MANET** vyberieme hodnotu *include:* **yes**. Nastavenia ukončíme pomocou „next“ a „finish“.

4. V ďalšom kroku naimportujeme obrázok na pozadie pomocou „View –> Background – > Add Image“. Zaklikneme „Import background image“,vo *file path* vyberieme cestu k obrázku „obr.tif“ a potvrdíme pomocou **OK.**
5. OM nás upozorní, že sa nachádzame v editačnom móde obrázku. Po kliknutí na obrázok sa zobrazia koncové body obrázka, pomocou ktorých je možné upravovať polohu a veľkosť obrázka. Spôsobom „drag and drop“ umiestnime ľavý dolný roh obrázka na bod **0, 0** a pravý horný roh na bod **350, 250.** Z editačného módu obrázka odídeme pravým kliknutím na obrázok a voľbou „Exit Map Editing Mode“. V prípade potreby sa do editačného módu dostaneme pomocou „ View –> Background –> Map Edit Mode“. Vzorové umiestnenie obrázku pozadia OM je na obr. 5.3.



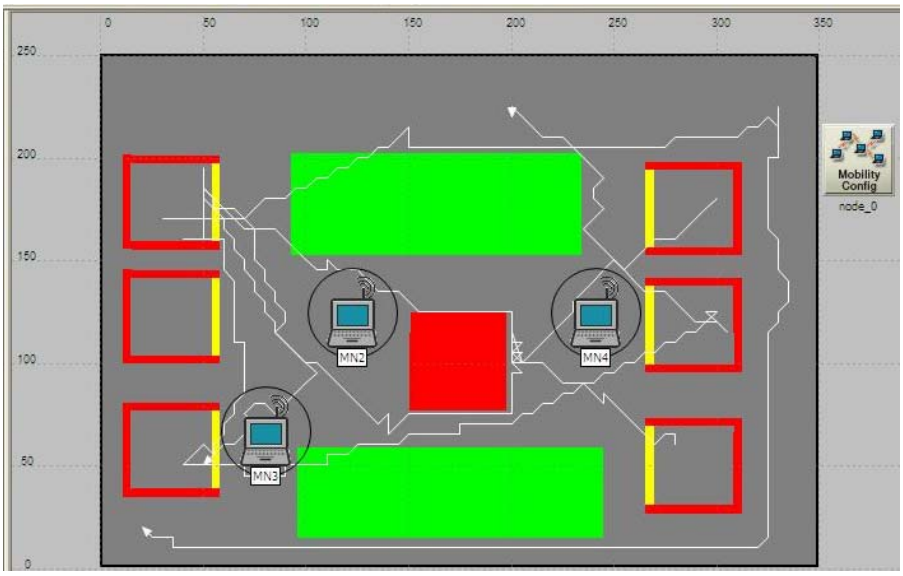
Obr. 5.3: Nastavenie pozadia projektu

6. Pomocou „Open Object Palette“ vložíme jeden objekt „Mobility Config“ a tri objekty „Manet Station“ typu „mobile node“. Objekty pomenujeme podľa obr. 5.4. Ich umiestnenie je iba názorné, pretože sú ich trajektórie fixné vzhľadom k podsieti.



Obr. 5.4: Vloženie objektov do projektu

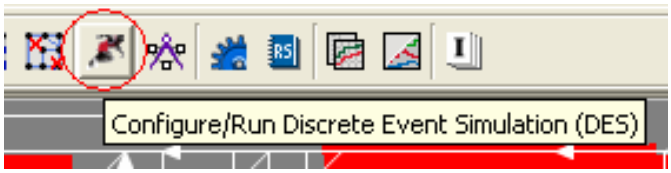
7. Projekt uložíme. Do domovskej zložky projektu nakopírujeme pripravené súbory s trajektóriami. Súbory pomenujeme podľa formátu „<project-scenario>-<node-name>.trj", teda pre stanicu **MN2** bude názov jej trajektórie: „TRJprojekt-scenario1-Campus Network.MN2.trj".
8. Stanica musí mať priradený profil z „Mobility config" pre možnosť použitia vlastného súboru s trajektóriou. So stlačeným „ctrl" označíme stanice, ktorým chceme priradiť profil mobility. Priradenie prevedieme pomocou menu „Topology –> Random Mobility –> Set Mobile Profile" a následne vyberieme „Mobility Profile name" **Default Random Waypoint**. V „Mobility config" nie je potrebné nastavovať žiadne parametre, pretože sú už prednastavené. Je však nutné, aby bolo vypnuté zaznamenávanie trajektórie. V opačnom prípade by sa nám prepísal súbor s trajektóriou novými hodnotami z náhodnej trajektórie. V danom profile je toto zaznamenávanie prvotne vypnuté.
9. Keď majú stanice nastavený profil, sú schopné využívať objekt „Mobility config". Pomocou neho je možné nastaviť vlastný súbor s trajektótiou. Priradíme trajektórie pomocou menu „Topology –> Random Mobility –> Set Trajectory Created from Random Mobility. Z kontextu je zrejmé, že OM nastaví staniciam súbory trajektórie vytvorené z náhodnej mobility. V týchto súboroch sa však nachádzajú dáta, ktoré sme vopred vygenerovali z MATLABu. Pre správne priradenie musia byť súbory pomenované v skôr spomenutom formáte. Po priradení sa v editore zobrazia trajektórie staníc vyznačené bielymi čiarami, čo je vidieť na obr. 5.5.



Obr. 5.5: Stanice s priradenými trajektóriami

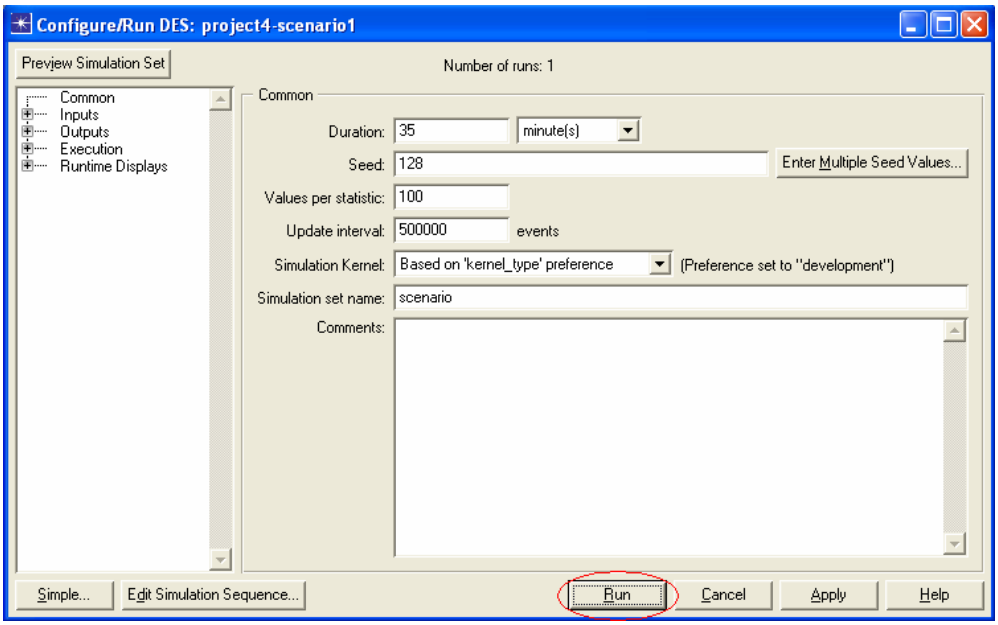
5.3 Simulácia modelu

Projekt je vytvorený a pripravený pre simuláciu. Pred samotnou simuláciou je však potrebné povoliť zaznamenávanie pohybu, aby bolo možné skontrolovať výsledný pohyb staníc. V menu „DES" zatrhneme „Record Node Movement 2D Animation for Subnet". Pohyb bude možné po simulácii sledovať pomocou prehrávača „2D player". Pre nastavenie simulácie klikneme na ikonu „Configure/Run Discrete Event Simulation" viď obr. 5.6.



Obr. 5.6: Ikona nastavenia simulácie

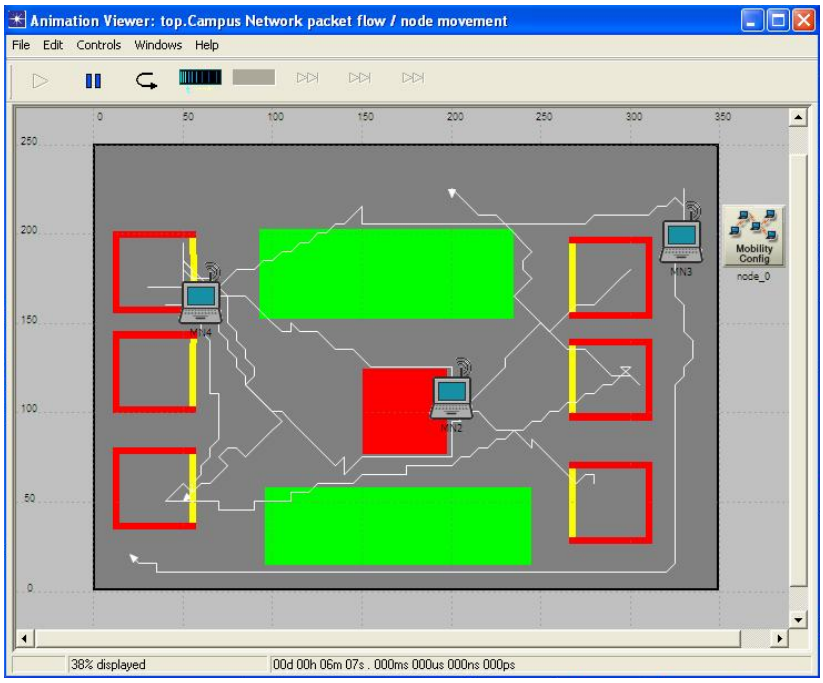
Čas simulácie nastavíme na hodnotu 35 minút. Ostatné polia ponecháme bez zmeny. Projekt odsimulujeme pomocou „RUN“. Nastavenia simulácie sú na obr. 5.7.



Obr. 5.7: Nastavenia simulácie

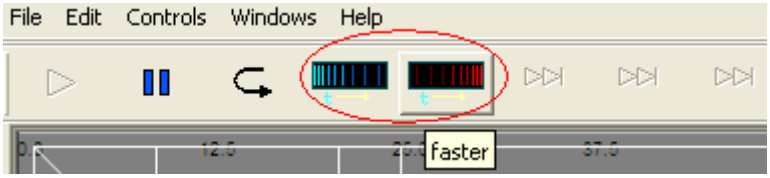
5.4 Overenie pohybu staníc v 2D player

Po odsimulovaní je možné overiť pohyb staníc pomocou prehrávača, ktorý počas simulácie zaznamenáva pohyb staníc v podsieti. Zvolíme menu „DES – Play 2D Animation. Po krátkom čase sa otvorí okno prehrávača a spustí sa prehrávanie viď obr. 5.8.

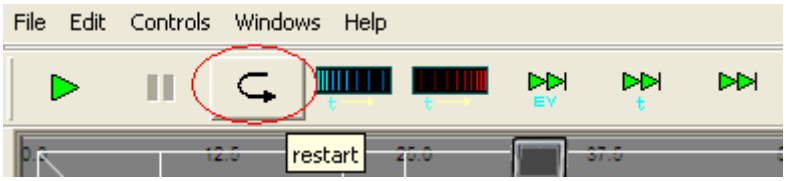


Obr. 5.8: 2D prehrávač

Rýchlosť prehrávania je možné meniť pomocou tlačidiel rýchlosti viď obr. 5.9. Reštart prehrávania je možný pomocou tlačidla obnovy viď obr. 5.10.



Obr. 5.9: Ovládanie rýchlosti prehrávania



Obr. 5.10: Reštart prehrávania

6 MATLAB Engine

Pre volanie MATLAB softwaru z programov napísaných v jazyku C/C++ slúži MATLAB Engine. Knižnica MATLAB Engine obsahuje programy, ktoré umožňujú volať funkcie MATLABu z externých programov. Pre použitie Enginu musí byť nainštalovaný MATLAB, avšak nie je možné spúšťať MATLAB Engine programy iba s použitím MATLAB kompilátora. Engine programy sú nezávislé na programoch v jazyku C/C++, ktoré oddelene komunikujú s MATLAB procesom cez komunikačné rozhranie. MATLAB poskytuje knižnicu funkcií, ktoré umožňujú spustiť a ukončiť MATLAB proces, zaslať dáta do MATLABu a späť a zaslať príkazy pre spracovanie v MATLABe.

6.1 ENGINE knižnica

V tab. 6.1 sú uvedené funkcie knihovne MATLAB Enginu pre použitie v jazyku C/C++.

Tab. 6.1: Funkcie knihovne MATLAB Engine

Funkcia	Účel
engOpen	spustenie MATLAB Engine
engClose	zastavenie MATLAB Engine
engGetVariable	získanie hodnôt z Enginu
engPutVariable	zaslanie hodnôt do Enginu
engEvalString	vykonanie MATLAB príkazu
engOutputBuffer	vytvorenie bufferu pre uloženie textového výstupu z MATLABu
engOpenSingleUse	spustenie MATLAB engine sekcie pre jednodielne nezdieľané použitie
engGetVisible	zistenie viditeľnosti sekcie MATLAB Engine
engSetVisible	nastavenie viditeľnosti sekcie MATLAB Engine

Príklady použitia Engine príkazov v C/C++ programe:

- Spustenie MATLAB enginu:

```
ep = engOpen(NULL);
```

- Vloženie hodnoty premennej „XP“ z jazyka C++ do premennej „A“ v MATLABe:

```
engPutVariable(ep, "A", XP);
```

- Vykonanie príkazu v MATLABe, konkrétne vykonanie funkcie s názvom „zapis_variable_func“ s jej vstupnými a výstupnými hodnotami:

```
engEvalString(ep,"[X_COORD,Y_COORD,P]=zapis_variable_func(A,B,C,D,E,F,G)
");
```

- Získanie hodnoty premennej „X_COORD“ z MATLABu a vloženie do premennej „XS“ v jazyku C++:

```
XS = engGetVariable(ep, "X_COORD");
```

6.2 Nastavenie ciest pre externé funkcie MATLABu

Pre správnu funkčnosť MATLAB Engine musia byť nastavené určité premenné. Pre využívanie externých funkcií a skriptov vytvorených v MATLABe musí mať MATLAB nastavené cesty k funkciám. Nastavenie ciest v MATLABe:

1. Pre nastavenie ciest v MATLABe treba kliknúť na „File -> Set Path…”.
2. Pomocou „Add Folder...“ sa priradí cesta k externej funkcii.
3. Pomocou „Save“ sa nastavenie uloží.

Správne nastavenie cesty k externej funkcii sa dá overiť napísaním funkcie do príkazového riadku MATLABu. Pri správnom nastavení sa daná funkcia spustí bez toho, aby bola cesta k danej funkcii nastavená v „Current Directory“

6.3 Registrácia MATLABu ako COM Server

Pre správne spúšťanie Enginu je potrebné mať používanú verziu MATLABu zaregistrovanú ako COM server. Táto registrácia je súčasťou inštalácie MATLABu a mala by byť automaticky prevedená pri inštalácii. Ak z nejakého dôvodu nedôjde k správnej registrácii, môže dôjsť k chybe pri spúšťaní Enginu. Týmto dôvodom môže byť inštalácia viacerých verzií MATLABu na jeden operačný systém. Manuálna registrácia sa prevedie nasledujúcimi príkazmi v príkazovom riadku systému:

```
cd matlabroot\bin\win32
matlab /regserver
```

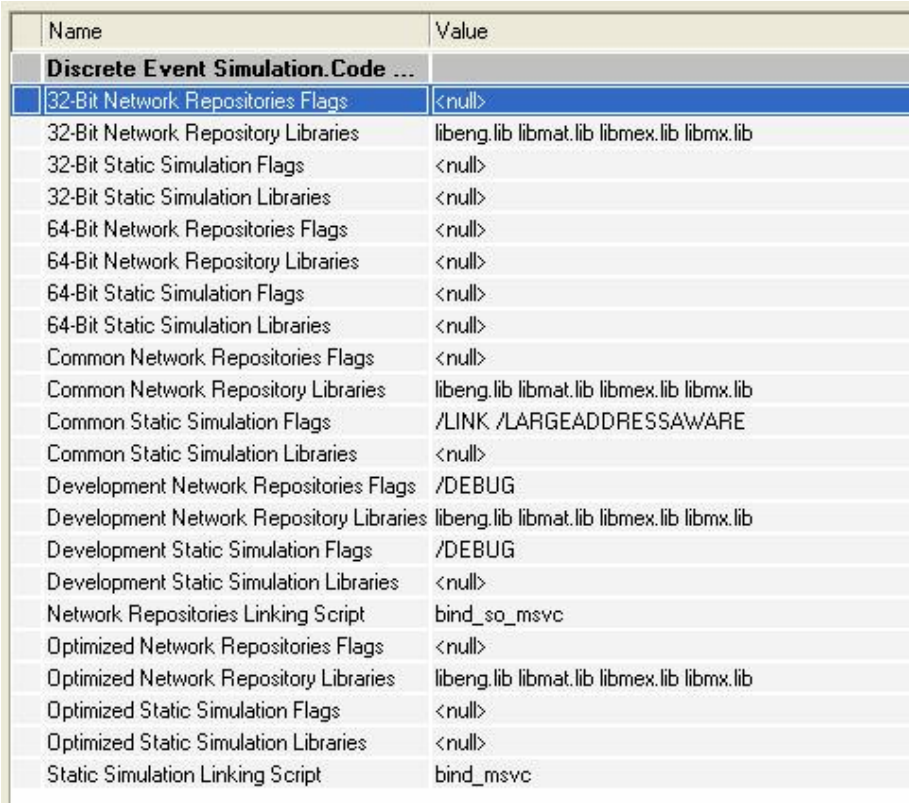
6.4 Nastavenie ciest pre OPNET/MATLAB rozhranie

V OM je potrebné nastaviť názvy potrebných knihovní Enginu. Prostredie preferencií OPNETu sa vyvolá kliknutím na „Edit -> Preferences“. Nastavenie premenných simulácie je pod preferenciami „Discrete Event Simulation -> Code Generation -> Linking“. Pre správne nastavenie je potrebné:

- vložiť do „32-Bit Network Repository Libraries“,
  „Common Network Repository Libraries“,
  „Developement Network Repository Libraries“,
  „Optimized Network Repository Libraries“

  názvy potrebných knihovní Enginu „libeng.lib libmat.lib libmex.lib libmx.lib“

  Vzorové nastavenie je zobrazené na obr. 6.1.



Name	Value
Discrete Event Simulation.Code ...	
32-Bit Network Repositories Flags	<null>
32-Bit Network Repository Libraries	libeng.lib libmat.lib libmex.lib libmx.lib
32-Bit Static Simulation Flags	<null>
32-Bit Static Simulation Libraries	<null>
64-Bit Network Repositories Flags	<null>
64-Bit Network Repository Libraries	<null>
64-Bit Static Simulation Flags	<null>
64-Bit Static Simulation Libraries	<null>
Common Network Repositories Flags	<null>
Common Network Repository Libraries	libeng.lib libmat.lib libmex.lib libmx.lib
Common Static Simulation Flags	/LINK /LARGEADDRESSAWARE
Common Static Simulation Libraries	<null>
Development Network Repositories Flags	/DEBUG
Development Network Repository Libraries	libeng.lib libmat.lib libmex.lib libmx.lib
Development Static Simulation Flags	/DEBUG
Development Static Simulation Libraries	<null>
Network Repositories Linking Script	bind_so_msvc
Optimized Network Repositories Flags	<null>
Optimized Network Repository Libraries	libeng.lib libmat.lib libmex.lib libmx.lib
Optimized Static Simulation Flags	<null>
Optimized Static Simulation Libraries	<null>
Static Simulation Linking Script	bind_msvc

Obr. 6.1: Nastavenie premenných OPNETu

6.5 Vytvorenie spúšťacieho „*.BAT“ súboru

Cesty k zložkám s knihovňami MATLAB Enginu je potrebné nalinkovať do systému. Funkčný a jednoduchý spôsob je vytvorenie spúšťacieho súboru, ktorý dynamicky nalinkuje potrebné cesty a následne spustí OM. Súbor je po jeho vytvorení možné používať na spúšťanie OM s nalinkovanými knihovňami MATLAB Enginu. Ako prvé je potrebné vytvoriť nový textový dokument s nasledujúcim obsahom:

```
call "C:\Program Files\Microsoft Visual Studio
9.0\Common7\Tools\vsvars32.bat"

@set PATH=C:\Program Files\MATLAB\R2010b\bin\win32;%PATH%
@set LIB=C:\Program Files\MATLAB\R2010b\extern\lib\win32\microsoft;%LIB%
@set INCLUDE=C:\Program Files\MATLAB\R2010b\extern\include;%INCLUDE%

cd C:\Program Files\OPNET\16.0.A\sys\pc_intel_win32\bin
set
modeler.exe
```

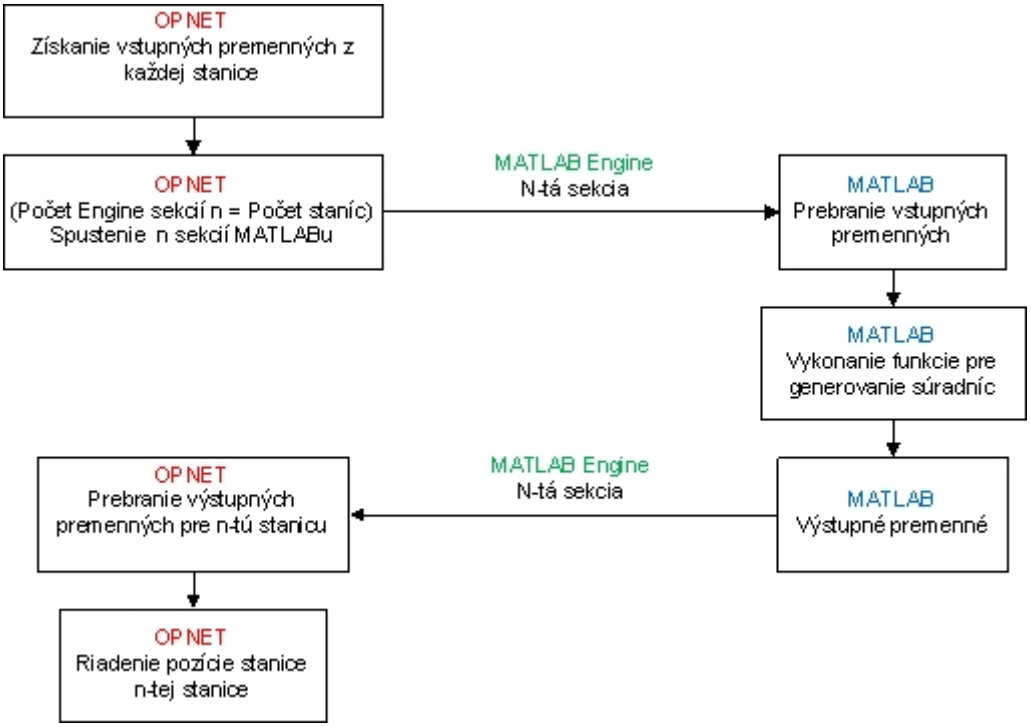
Následne sa premenovaním textového dokumentu na „opnet.bat“ vytvorí spúšťací „BAT“ súbor. Nastavenie cesty pre MATLAB sa prevedie príkazom „`@set PATH=`“. Cesta pre knihovne je nastavená príkazom „`@set LIB=`“ a cesta pre hlavičkový súbor „engine.h“ príkazom „`@set INCLUDE=`“. Cesty sú závislé od použitej verzie MATLABu. Pri nastavovaní cesty k MATLABu pomocou tohto návodu je potrebné zmazať staticky nastavené cesty v premenných systému v položke „PATH“. V prípade viacerých verzií MATLABu sa predíde k zlému nastaveniu cesty z inej verzie, ktorá nie je používaná ako primárna.

7 Projekt s automatizovaným procesom

Ako už bolo spomenuté v kapitole 3, pozíciu je možné riadiť z externého zdroja súradníc pomocou priradenia vytvorenej trajektórie stanici, alebo priamou manipuláciu pozície stanice. V kapitole 4 je prezentovaný projekt, kde je stanici priradená trajektória vygenerovaná pomocou vytvorenej funkcie v MATLABe. Tento spôsob riadenia pozície však neumožňuje celý proces zautomatizovať tak, aby nebol potrebný zásah užívateľa od vytvorenia súboru s trajektóriou po samotnú simuláciu. Automatizácia zlyháva v bode, kde je potrebné trajektóriu priradiť stanici počas simulácie. To však súčasná verzia OM nepodporuje podľa odbornej odpovede z podpory OM. Trajektóriu je možné priradiť pred simuláciou a následne projekt odsimulovať. Celý proces je však možné zautomatizovať s využitím priamej manipulácie pozície stanice. Priama manipulácia pozície stanice je popísaná v kapitole 3.3.

7.1 Model automatizovaného procesu

V projekte s automatizvaným procesom sa jedná o proces predávania hodnôt medzi OM a MATLABom počas simulácie bez zásahu užívateľa. MATLABu sú počas simulácie predané vstupné hodnoty z premenných OM v jazyku C++ a následne sa vykoná zavolanie externej funkcie pre vygenerovanie trajektórie. Po vykonaní funkcie sa predajú výstupné hodnoty z MATLABu späť do OM. Na základe výstupných hodnôt sa priamou manipuláciou riadi pozícia stanice. Na obr. 7.1 je zobrazený diagram automatizovaného procesu.



Obr. 7.1: OPNET MATLAB spolupráca

Vstupy a výstupy funkcie sú popísane v kapitole 4.6. Tie sú pomocou MATLAB Engine spracované v jazyku C++. Spolupráca MATLABu s OM je popísaná v kapitole 6.

7.2 Vytvorenie projektu

Nový projekt je možné vytvoriť podľa úvodného návodu z kapitoly 5.2 po piaty bod. Meniť sa budú iba rozmery pracovnej plochy podľa vstupného obrázka, ktorý si definuje užívateľ. V tomto projekte sa však použije iný druh riadenia pozície stanice a samotná úprava stanice. Po vytvorení projektu je vhodné vložiť na pozadie vstupný obrázok, ktorý bude otočený v ose Y. Je to z toho dôvodu, že OPNET verzie 16 zobrazuje Y os opačne, než MATLAB pri vykreslení jeho výstupu. Toto však na funkčnosť nemá žiadny vplyv. Vkladanie obrázka na pozadie pracovnej plochy OPNETu je tiež popísané v kapitole 5.2.

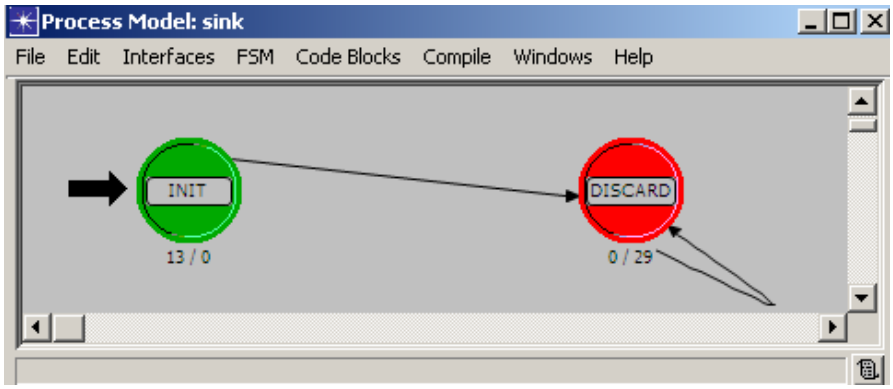
7.2.1 Vytvorenie duplikátu stanice

Pri zámere úpravy pôvodných modelov, je vhodné vytvoriť ich duplikáty a tie následne upravovať. V opačnom prípade by sa úpravy zapísali do pôvodných modelov. Tie by boli v prípade chybnej úpravy nefunkčné. Duplikáty sa ukladajú do vlastnej zložky pod vlastným názvom. Po kliknutí na „Open Object Palette“ sa otvorí okno s paletou objektov. Tu bol zvolený model pre úpravu „wlan_wkstn_adv“ typu „Mobile Node“. Po kliknutí na „Model Details“ sa zobrazia detaily modelu. Tu je možné model derivovať pomocou „Derive New“. Pre podporu iba mobilnej stanice treba zvoliť „NO“ v okne „Node Types“ u položky „fixed “. Vytvorený duplikát sa uloží ako derivovaný objekt pomocou „Save“ do vlastnej zvolenej zložky. V tomto projekte bol použitý názov duplikátu „engine_ wlan_wkstn_adv“.

Túto duplikovanú stanicu je možné vyhľadať v palete objektov. Po vložení stanice do pracovného prostredia je potrebné vstúpiť do „Node Model“ a uložiť stanicu na tejto úrovni pomocou „File -> Save“ do zložky s duplikátom. V tomto projekte bol použitý názov modelu „engine_ wlan_wkstn_adv“. Ďalej je potrebné duplikátu priradiť vytvorený „Node Model“. To sa prevedie v menu „Edit Attributes“ stanice pri zatrhnutej voľbe „Advanced“. V kolónke „model“ treba vybrať vytvorený model. Po aplikovaní uvedených krokov sa vytvorí kompletný duplikát stanice. Po jeho úprave sa nebudú zmeny zapisovať do pôvodného modelu.

7.2.2 Vytvorenie vlastného procesoru „engine“

Zdrojový kód, ktorým sa riadi pohyb stanice sa nachádza v samostatnom procesore. Ten sa vytvorí pomocou „Create Proccessor“. Vytvorený procesor je základný s názvom „sink“. Pre potrebu úpravy je potrebné procesor uložiť pod vlastným názvom do vlastnej zložky obdobne ako v kapitole 7.2.1. V tomto projekte bol použitý názov „engine“. V „Edit Attributes“ vytvoreného procesoru je potrebné daný uložený procesor priradiť v kolónke „process model“. Neupravený procesor je na obr. 7.2. Stavy a prechody medzi nimi sú popísané v literatúre [10].



Obr. 7.2: Procesor sink

7.3 Vytvorenie atribútov engine mobility

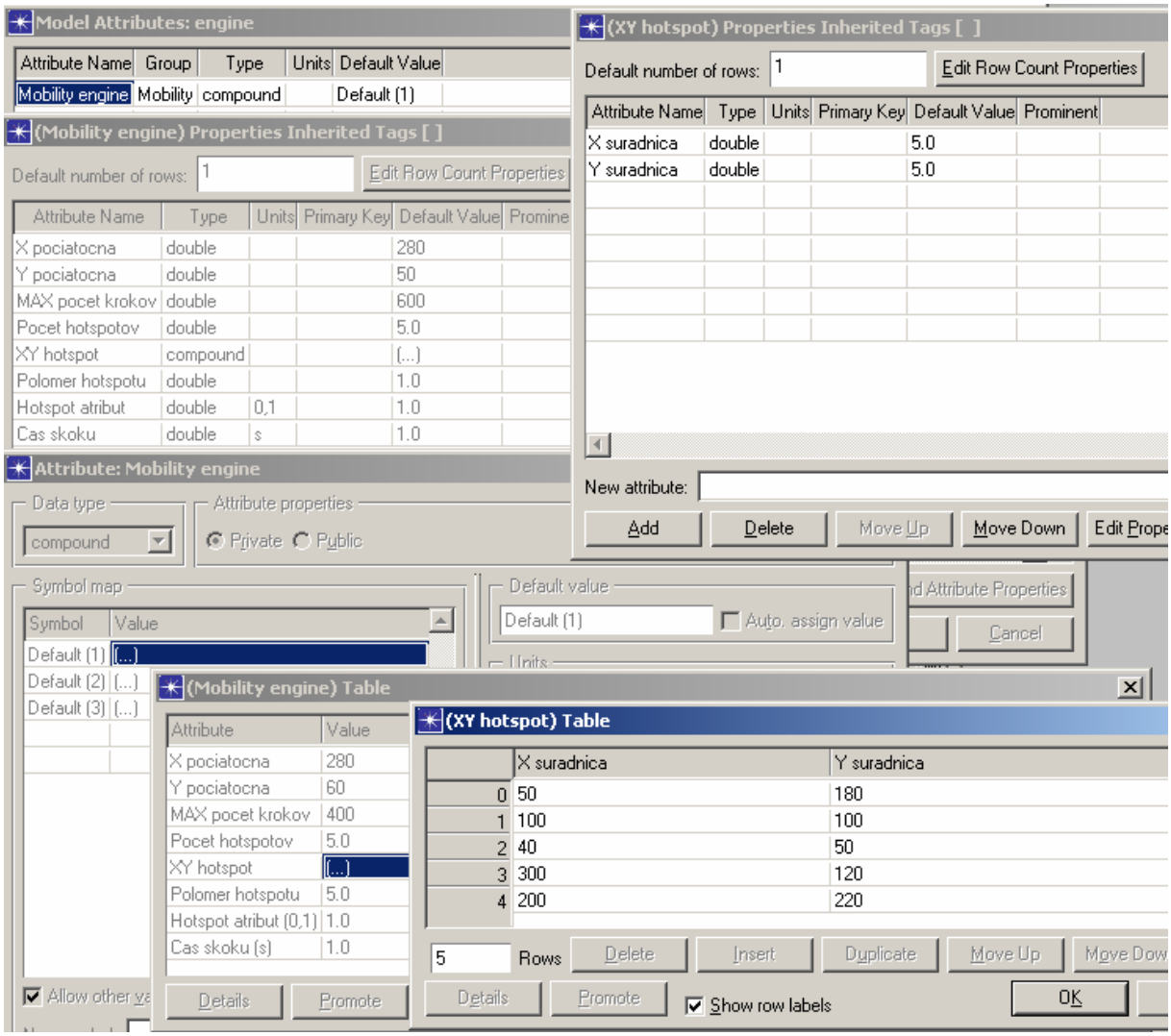
Pre editovanie vstupných premenných, ktoré bude jednoduché pre užívateľa, treba vytvoriť vhodné prostredie na to určené. Týmto prostredím je vytvorenie vlastných atribútov stanice, do ktorých sa budú vkladať potrebné hodnoty. Jedná sa o grafické rozhranie, ktoré je užívateľovi bližšie, ako editovanie samotného kódu. Predíde sa tým aj možnému chybnému editovaniu zdrojového kódu do nefunkčnej podoby. Dané hodnoty sa z grafického rozhrania vyčítajú do zdrojového kódu, kde sú ďalej spracovávané.

V projekte bola vytvorená skupina atribútov s názvom „Mobility engine“. Táto skupina obsahuje vstupné premenné, ktoré sú predávané vytvorenej funkcii v MATLABe. Atribúty skupiny sú nasledovné:

- X počiatočná - počiatočná X súradnica
- Y počiatočná - počiatočná Y súradnica
- MAX počet krokov - maximálny počet krokov pre generovanie súradníc
- Počet hotspotov - počet použitých hotspotov
- XY hotspot - spoločné pole súradníc pre hotspoty
  - Number of Rows - celkový počet hotspotov
  - X súradnica - X súradnica pre daný hotspot
  - Y súradnica - Y súradnica pre daný hotspot
- Polomer hotspotu - vzdialenosť od hotspotu pre jeho dosiahnutie
- Hotspot atribút - atribút zapnutia vyhľadávania hotspotov, 1 pre zapnutie, 0 pre vypnutie
- Čas skoku - čas medzi skokom stanice z pozície A na pozíciu B

7.3.1 Spôsob vytvorenia atribútov „Mobility engine“

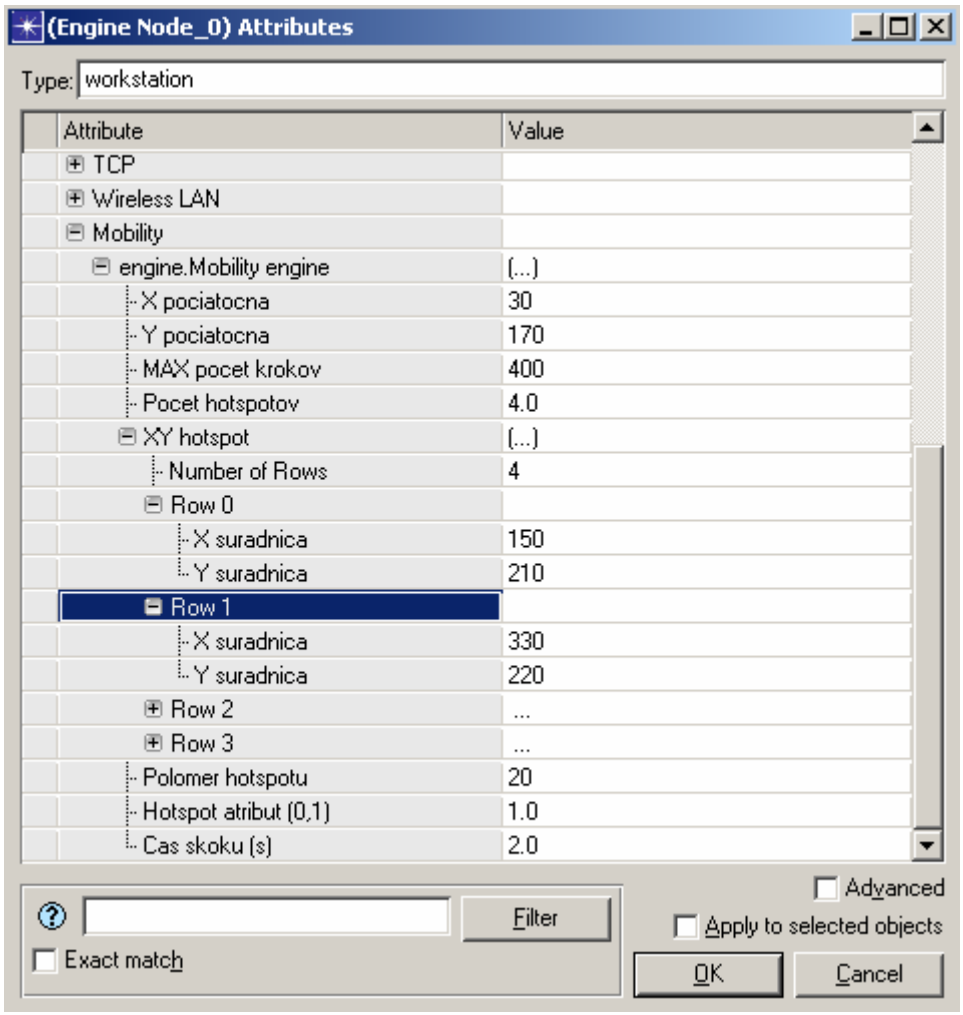
Spôsob vytvárania atribútov je popísaný v literatúre [10]. Stanici boli vytvorené atribúty popísané v kapitole 7.3. Atribút „Mobility engine“ je typu „compound“, čo znamená, že sa jedná o vnorené menu s názvom skupiny „Mobility“. V tomto menu sa nachádzajú jednotlivé atribúty. Vytváranie atribútov cez menu „Interfaces -> Model Attributes“ je zobrazené na obr. 7.3. Na obr. 7.4 sú zobrazené atribúty stanice v menu „Edit Attributes“ po ich vytvorení.



Obr. 7.3: Vytváranie atribútov stanice

Popis okien v obr. 7.3:

- **Model attributes: Engine** – zobrazenie prvej vrstvy menu atribútov, ktorá je typu „compound“ a je teda rodičom vlastného menu.
- **(Mobility engine) Properties Inherited Tags** [] – zobrazenie vlastného menu atribútu „Mobility engine“, v ktorom sa nachádzajú atribúty nižšej vrstvy.
- **(XY hotspot) Properties Inherited Tags** [] – zobrazenie menu atribútu „XY hotspot“, ktorý je v menu na najnižšej úrovni. Tento atribút obsahuje hodnoty hotspotov.
- **Attribute: Mobility engine** – okno pre editovanie vlastností položiek v menu atribútu „Mobility engine“. V kolónke „Symbol map“ sú zadefinované 3 profily s preddefinovanými hodnotami atribútov. V nastavení atribútov stanice stačí vybrať príslušný profil a všetky hodnoty sa automaticky doplnia hodnotami z daného profilu.
- **(Mobility engine) Table** – okno s tabuľkou hodnôt pre profil „Default (1)“.
- **(XY hotspot) Table** – tabuľka hodnôt hotspotov atribútu „XY hotspot“ pre profil „Default (1)“.



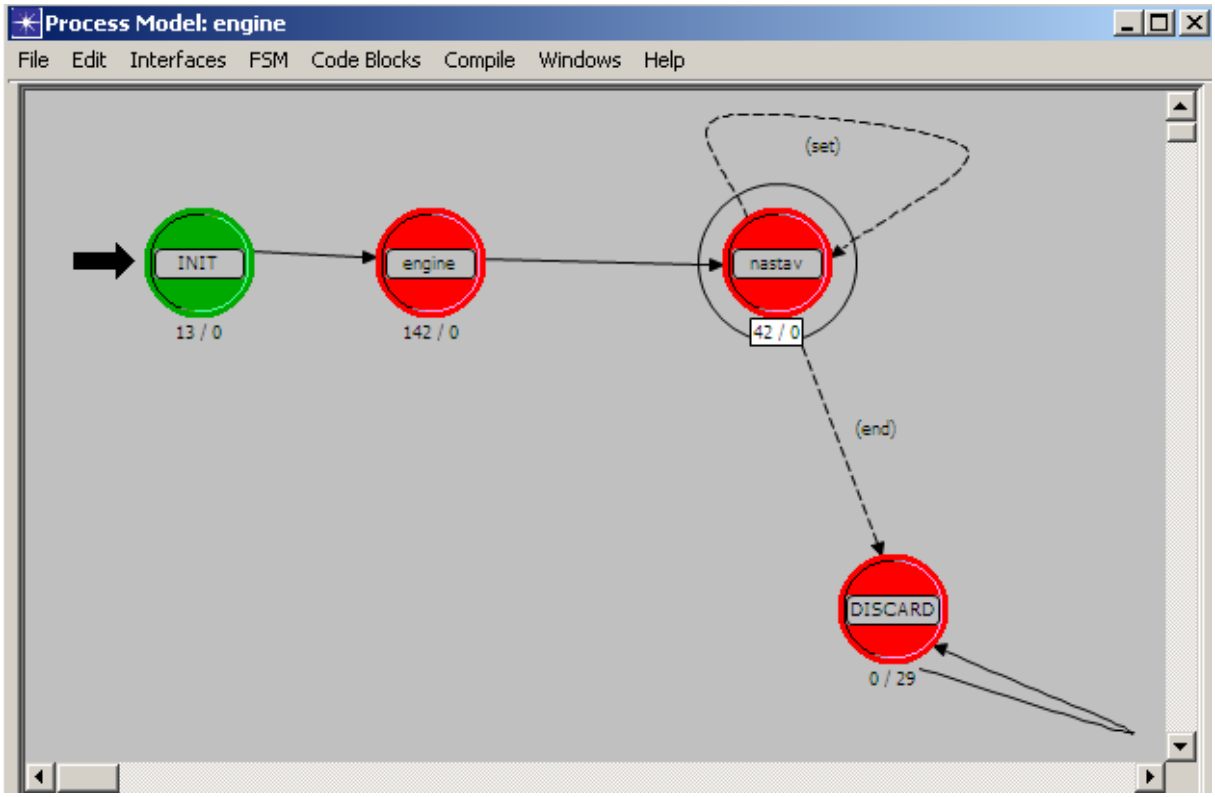
Obr. 7.4: Vytvorené atribúty stanice

7.4 Vloženie vlastného kódu pre zautomatizovanie procesu

Na obr. 7.5 je zobrazený upravený procesor s názvom „engine“. Boli v ňom vytvorené dva nové stavy „engine“ a „nastav“. V stave „engine“ sa vyčítajú premenné z atribútov stanice, ktoré sa následne spracujú. Prevedie sa volanie MATLAB funkcie a spracujú sa výstupné premenné. Zdrojový kód je zobrazený v prílohe B. Stav „nastav“ obsahuje kód, ktorý vykonáva samotné nastavovanie atribútov pozície stanice. Nastavovanie trvá, pokiaľ sa nevyčerpajú súradnice. Zdrojový kód tohto stavu je zobrazený v prílohe C. Všetky potrebné premenné sú definované v bloku „Header blok“, ktorý je zobrazený v prílohe D. Ďalšie premenné boli definované v bloku „State Variables“ Tieto sú:

```
Objid \parent_obj_id;
Objid \my_obj_id;
bool  \set;
bool  \end;
bool  \init;
bool  \end_traj;
Objid \mobility_engine_obj_id;
Objid \engine_mobility_engine_obj_id;
```

```
Objid \node_atr_obj_id;
Objid \hot_mobility_engine_obj_id;
Objid \poz_hot_mobility_engine_obj_id;
```



Obr. 7.5: Engine procesor

Komentáre sa nachádzajú priamo v zdrojových kódoch.

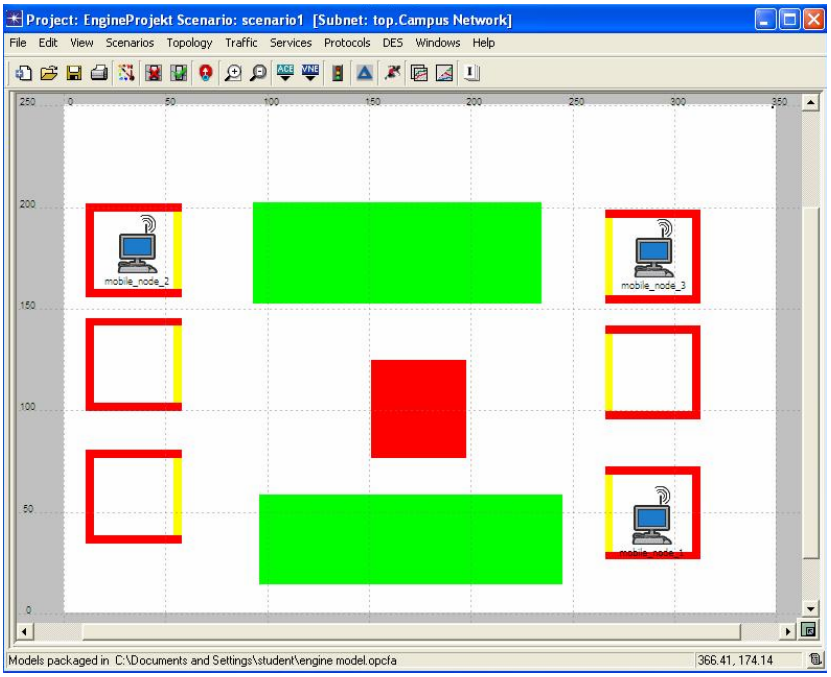
7.5 Simulácia a overenie výsledkov.

Pred simuláciou je potrebné nastaviť hodnoty vytvorených atribútov v skupine „Mobility“ pre každú stanicu. Atribúty sú popísané v kapitole 7.3. Menu atribútov stanice sa vyvolá pomocou „Edit Attributes“. Vzorové nastavenie hodnôt staníc je v tabuľke tab. 7.1. Hodnoty boli zvolené testovaním tak, aby reprezentovali reálny pohyb stanice po mape na základe možnej reálnej situácie. Z výsledného pohybu je tak vidieť, že sa stanice pohybujú medzi budovami, obchádzajú prekážky a vstupujú do a vystupujú z budov.

Tab. 7.1: Hodnoty atribútov

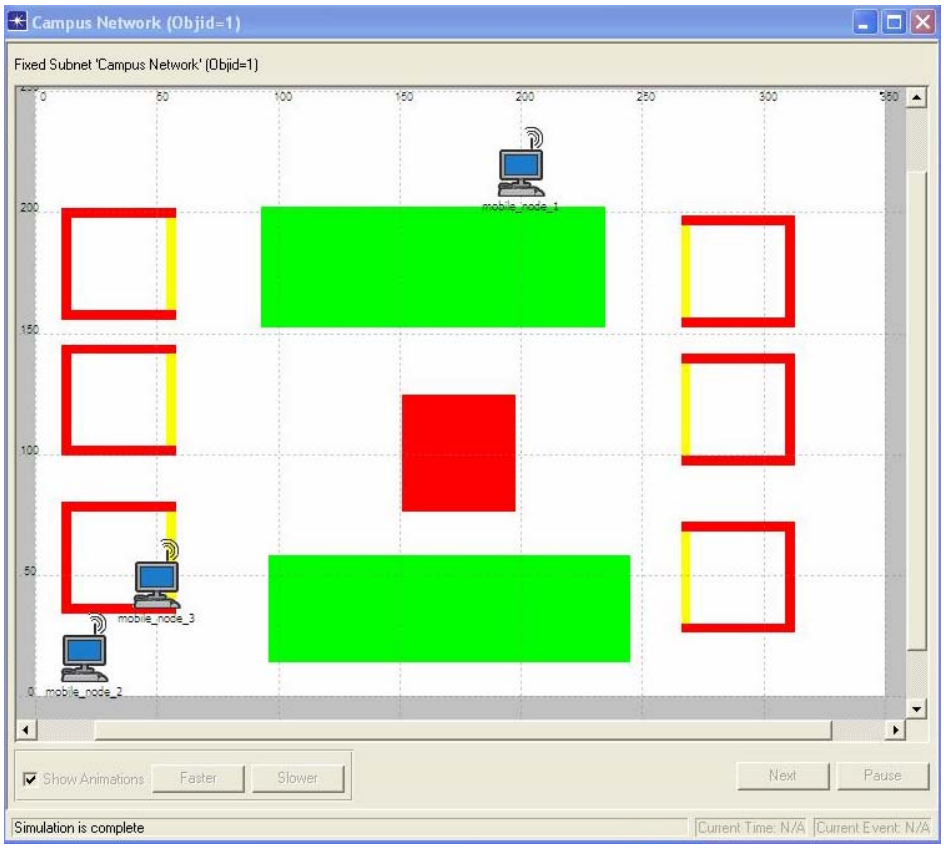
Atribúty / Stanica	Engine Node	Engine Node 0	Engine Node 1
X počiatočná	280	30	300
Y počiatočná	60	170	180
MAX počet krokov	400	400	400
Počet hotspotov	5	4	3
XY hotspot - Number of Rows	5	4	3
X súradnica Row 0 Row 1 Row 2 Row 3 Row 4	 50 100 40 300 200	 150 330 320 20 -	 140 50 50
Y súradnica Row 0 Row 1 Row 2 Row 3 Row 4	 180 100 50 120 220	 210 220 10 20 -	 70 190 50
Polomer hotspotu	5	5	5
Hotspot atribút	1	1	1
Čas skoku	1	2	3

Nakoľko boli vytvorené 3 profily hodnôt atribútov, stačí tieto profily staniciam priradiť namiesto vypisovania každej hodnoty zvlášť. Hodnoty atribútov sa nastavujú cez menu stanice „Edit Attributes“ v kolónke „Mobility“. Zobrazenie projektu je na obr. 7.6.



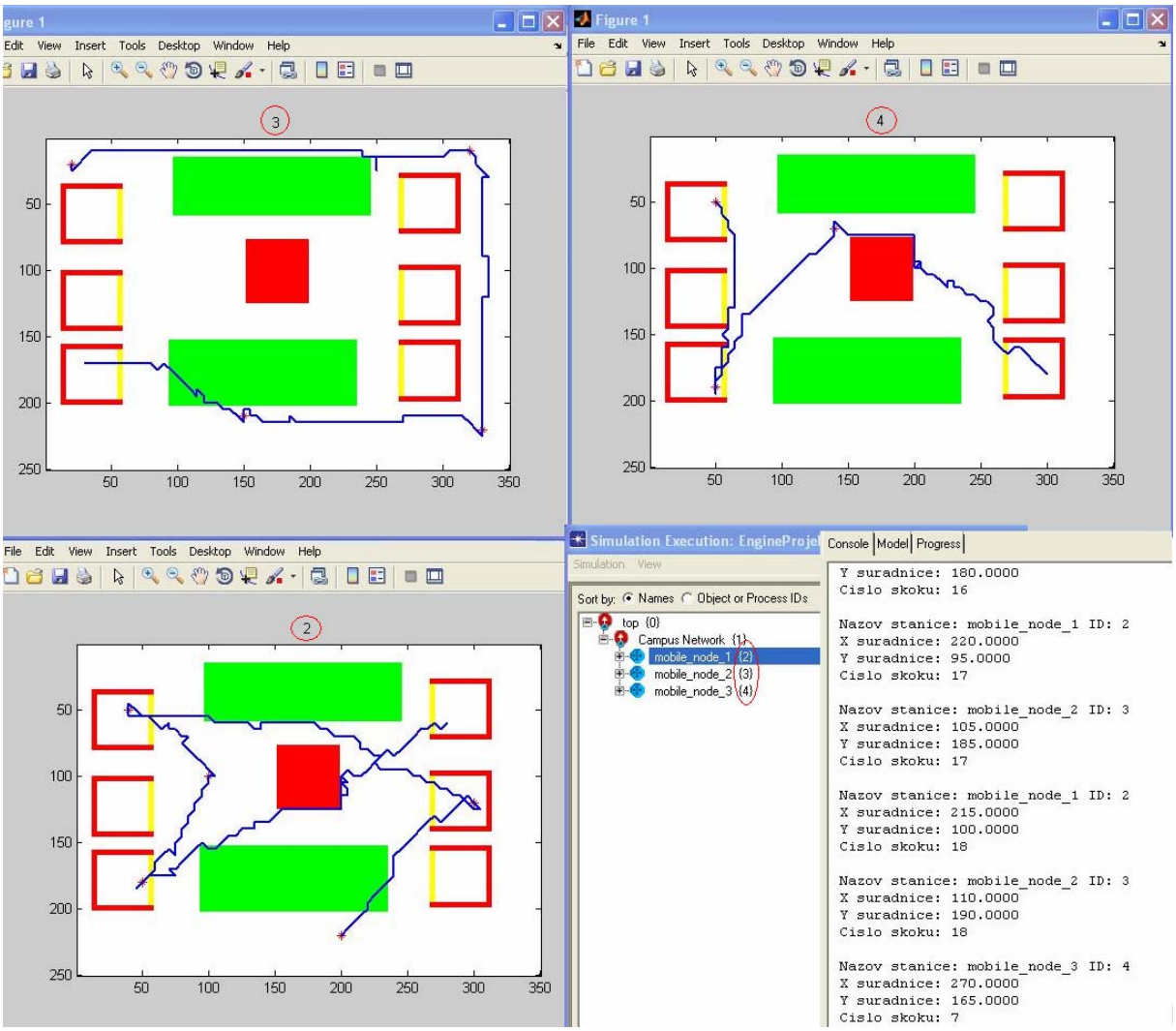
Obr. 7.6: Vytvorený projekt

Pre konfiguráciu simulácie je treba zvoliť „Configure/Run Discrete Event Simulation“. Čas simulácie treba nastaviť intuitívne podľa nastavenia vytvorenej mobility a jej hodnôt tak, aby stanice stihli vykonať pohyb po ich trajektóriach. Ak má stanica nastavený maximálny počet skokov 400 a čas skoku 3 sekundy, postačí nastaviť dobu simulácie na 20 minút. Ďalej bol povolený OPNET Debugger zvolený v záložke „Execution - OPNET Debugger“ zaškrtnutím položky „Use OPNET Simulation Debugger“. Pomocou výpisov debuggeru je možné kontrolovať stav simulácie. Simulácia sa spustí tlačidlom „Run“. Pre sledovanie pohybu staníc v priebehu simulácie treba zaškrtnúť voľbu „Show Animations“ v OSD záložke „Model“. V záložke „Model“ je rovnako potrebné mať zobrazenú vrstvu, na ktorej sa nachádzajú stanice, teda vnútri podsiete. Rýchlosť priebehu simulácie je možné meniť tlačidlami „Faster“ a „Slower“. Simulácii sa potvrdí pokračovanie cez „Simulation - > Continue“ Na obr. 7.7 je zobrazené grafické okno, ktoré zobrazuje pohyb staníc počas simulácie.



Obr. 7.7: Okno animácie

Na obr. 7.8 sú zobrazené výstupné okná MATLABu, na ktorých je zobrazený vstupný obrázok s vykreslenou trajektóriou. Nad každým obrázkom sa nachádza číslo, ktoré reprezentuje ID (identifikátor) stanice. ID sú zvýraznené červeným krúžkom. V obrázku napravo dole je vidieť v červenom krúžku ID každej stanice a textový výstup debuggeru. Rozdeľovanie ID čísiel objektom začína od čísla 1. Číslo jedna však pripadne sieti, v ktorej na stanice nachádzajú, teda „Campus Network“. Čísla 2, 3 a 4 sú potom v poradí priradené staniciam, ktoré sa v danej sieti nachádzajú.



Obr. 7.8: Okno debugeru a výstupov z MATLABu

V priebehu simulácie sa do konzoly vypisujú úkony, ktoré boli vykonané. Vo fáze inicializácie sa pre každú stanicu postupne vypisuje názov stanice, počet hotspotov, hodnoty hotspotov, ID stanice, informácia o vygenerovaní súradníc pre danú stanicu, počiatočné súradnice a ich celkový počet. Inicializácia staníc je zobrazená v nasledujúcom texte:

```
_____________ OPNET Simulation Debugger _____________

Type 'help' for Command Summary

ODB> continue

 Nazov stanice: mobile_node_1
 Pocet hotspotov stanice mobile_node_1 je: 5
 1 Hotspot - suradnica X stanice mobile_node_1 je: 50.00
 1 Hotspot - suradnica Y stanice mobile_node_1 je: 180.00
 2 Hotspot - suradnica X stanice mobile_node_1 je: 100.00
 2 Hotspot - suradnica Y stanice mobile_node_1 je: 100.00
 3 Hotspot - suradnica X stanice mobile_node_1 je: 40.00
 3 Hotspot - suradnica Y stanice mobile_node_1 je: 50.00
 4 Hotspot - suradnica X stanice mobile_node_1 je: 300.00
 4 Hotspot - suradnica Y stanice mobile_node_1 je: 120.00
 5 Hotspot - suradnica X stanice mobile_node_1 je: 200.00
```

```
5 Hotspot - suradnica Y stanice mobile_node_1 je: 220.00
ID stanice mobile_node_1 je: 2
Vygeneroval sa subor trj a suradnice
Pociatocne X suradnice: 280.0000
Pociatocne Y suradnice: 60.0000
Celkovy pocet suradnic: 220.0000

Nazov stanice: mobile_node_2
Pocet hotspotov stanice mobile_node_2 je: 4
1 Hotspot - suradnica X stanice mobile_node_2 je: 150.00
1 Hotspot - suradnica Y stanice mobile_node_2 je: 210.00
2 Hotspot - suradnica X stanice mobile_node_2 je: 330.00
2 Hotspot - suradnica Y stanice mobile_node_2 je: 220.00
3 Hotspot - suradnica X stanice mobile_node_2 je: 320.00
3 Hotspot - suradnica Y stanice mobile_node_2 je: 10.00
4 Hotspot - suradnica X stanice mobile_node_2 je: 20.00
4 Hotspot - suradnica Y stanice mobile_node_2 je: 20.00
ID stanice mobile_node_2 je: 3
Vygeneroval sa subor trj a suradnice
Pociatocne X suradnice: 30.0000
Pociatocne Y suradnice: 170.0000
Celkovy pocet suradnic: 186.0000

Nazov stanice: mobile_node_3
Pocet hotspotov stanice mobile_node_3 je: 3
1 Hotspot - suradnica X stanice mobile_node_3 je: 140.00
1 Hotspot - suradnica Y stanice mobile_node_3 je: 70.00
2 Hotspot - suradnica X stanice mobile_node_3 je: 50.00
2 Hotspot - suradnica Y stanice mobile_node_3 je: 190.00
3 Hotspot - suradnica X stanice mobile_node_3 je: 50.00
3 Hotspot - suradnica Y stanice mobile_node_3 je: 50.00
ID stanice mobile_node_3 je: 4
Vygeneroval sa subor trj a suradnice
Pociatocne X suradnice: 300.0000
Pociatocne Y suradnice: 180.0000
Celkovy pocet suradnic: 139.0000
```

Po inicializácii nasleduje fáza nastavovania súradníc. Do konzoly sa postupne podľa času simulácie vypisuje názov stanice, jej aktuálne nastavené hodnoty súradníc a číslo skoku stanice na nové súradnice. Výpis simulácie bez použitia krokovania je nasledovný:

```
Nazov stanice: mobile_node_1 ID: 2
X suradnice: 280.0000
Y suradnice: 60.0000
Cislo skoku: 1

Nazov stanice: mobile_node_2 ID: 3
X suradnice: 30.0000
Y suradnice: 170.0000
Cislo skoku: 1

Nazov stanice: mobile_node_3 ID: 4
X suradnice: 300.0000
Y suradnice: 180.0000
Cislo skoku: 1

Nazov stanice: mobile_node_1 ID: 2
X suradnice: 275.0000
Y suradnice: 65.0000
Cislo skoku: 2
```

```
 Nazov stanice: mobile_node_2 ID: 3
 X suradnice: 35.0000
 Y suradnice: 170.0000
 Cislo skoku: 2
```

V prípade krokovania simulácie obsahuje výpis podrobnejšie informácie:

```
_____________________________ (ODB 16.0.A: Event) _____________________________

    * Time   :  31.0 sec, [31s]
    * Event  :  execution ID (247), schedule ID (#257), type (self intrpt)
    * Source :  execution ID (245), top.Campus Network.mobile_node_1.engine
[Objid=606] (processor)
    * Data   :  code (0)
    > Module :  top.Campus Network.mobile_node_1.engine [Objid=606]
(processor) [process id: 13]

ODB> next

 Nazov stanice: mobile_node_1 ID: 2
 X suradnice: 200.0000
 Y suradnice: 105.0000
 Cislo skoku: 32
```

Po vyčerpaní súradníc sa ukončí nastavovanie súradníc pre danú stanicu a vypíše sa informácia o ukončení:

```
Nazov stanice: mobile_node_1 ID: 2
Koniec suradnic
```

8 Záver

Cieľom práce bolo zoznámiť sa so simulačným nástrojom OPNET Modeler so zameraním na možnosti definície trajektórie pohybu bezdrôtovej stanice. Pri simulácii komunikačných sietí sa stáva riadenie pohybu staníc alebo celých podsietí kľúčovým. Model bezdrôtovej komunikácie v simulačnom prostredí OM umožňuje pri zachovaní funkcionality staníc a podsietí meniť ich polohu založenú na náhodne vybraných cestách alebo vopred definovaných trajektóriach.

V práci sú detailne preskúmané a teoreticky spracované stávajúce možnosti riadenia pohybu stanice v prostredí OM. Medzi tieto možnosti patrí konfigurácia náhodnej mobility, pričom sa jedná o implementáciu riadenia pohybu stanice na základe konfigurácie náhodnej mobility pomocou objektu „Moblity config“. Pri spôsobe riadenia pomocou tohto objektu sa jedná o súvislý pohyb, pretože sa o to stará priamo jadro OM. Tento objekt umožňuje vytvárať rôzne profily s nastavením parametrov náhodnej mobility a následne ich priradiť mobilným staniciam. Ďalší spôsob je doplnenie riadenia pohybu stanice pomocou náhodnej mobility o možnosť riadenia z externého vstupu. Vďaka tomu je stanica riadená priamo užívateľom, keďže sa jedná o trajektóriu definovanú užívateľom. Medzi možnosti riadenia patrí aj priama manipulácia pozície stanice. Pozícia stanice je riadená nejakým procesom, pričom sa nejedná o súvislý pohyb, pretože sú hodnoty pozície diskrétne.

Pomocou nástroja MATLAB bola vytvorená funkcia pre spracovanie mapového podkladu v podobe bitmapového obrázku do takej podoby, aby bola použiteľná v simulačnom prostredí OPNET Modeler pre riadenie pohybu bezdrôtovej stanice. Pre realizáciu automatizovaného procesu od vytvorenia trajektórie po následné riadenie pohybu stanice bola použitá knihovňa MATLAB Engine, ktorá umožňuje volanie MATLAB funkcií z jazyka C/C++. Na tomto jazyku je založený OM. V kombinácii s MATLAB Engine bolo pre riadenie pohybu použité riadenie pomocou priamej manipulácie pozície stanice, ktoré umožňuje automatizáciu modelu. Použitie riadenia pohybu stanice pomocou súborov s trajektóriami v automatizovanom modeli zlyháva v bode, kde je potrebné priradiť trajektórie staniciam počas simulácie. Toto súčastná verzia OM neumožňuje.

V práci je popísané vytvorenie dvoch modelov bezdrôtovej siete v ktorých bolo použité riadenie pohybu stanice na základe skôr spracovaných mapových podkladov pomocou nástroja MATLAB. Prvý model reprezentuje riadenie pozície stanice pomocou súborov s trajektóriami. Jedná sa o neautomatizovaný proces, kde treba najskôr vytvoriť súbory s hodnotami a následne scenár odsimulovať. Druhý model reprezentuje projekt s automatizovaným procesom od vygenerovania súradníc trajektórii po odsimulovanie projektu bez zásahu užívateľa. V práci bolo overené, že sa stanice v OM pohybujú po rovnakej trajektórii, ktorá bola vygenerovaná v MATLABe. Všetky postrehy a výsledky boli v práci podrobne zdokumentované.

Literatúra

[1] ILYAS, M.: The Handbook of Ad Hoc Wireless Networks. Boca Raton: CRC Press, 2003, ISBN:0-8493-1332-5.

[2] Answers. *Mobile ad hoc network* [online]. c2008, [2010-12-2].
Dostupné z: <http://www.answers.com/topic/mobile-ad-hoc-network>

[3] OPNET Technologies, OPNET Modeler Product Documentation Release 16.0, 2010.

[4] MOORE, H.: MATLAB for Engineers. New York: Prentice Hall, 2008, ISBN: 978-0136044222.

[5] MATLAB, MATLAB Help version 7.5, R2007b.

[6] AIT ALI, Kathina; MUSTAPHA LALAM, Mustap; MOALIC Laurent; BAALA, Oumaya. 2010 Ninth International Conference on Networks. *V-MBMM: Vehicular Mask-based Mobility Mode.*

[7] T. Camp, J. Boleng, and V. Davies. *A survey of mobility models for ad hoc network research*, Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, no. 5, pp. 483–502, 2002.

[8] A. Mahajan, N. Potnis, K. Gopalan, and A.-I. A. Wang. *Urban mobility models for vanets*, Proc of the 2nd IEEE International Workshop on Next Generation Wireless Networks, December, 2006.

[9] M. Fiore, J. Härri, F. Filali, and C. Bonnet. *Vehicular Mobility Simulation for VANETs*, Proc of the 40th IEEE Annual Simulation Symposium, March, 2007.

[10] O. Zeman. *Implementace simulačního modelu zjednodušené databáze diffserv-mib*. Diplomová práce, Brno: FEKT VUT v Brne, 2008. 55 s.

Zoznam použitých skratiek

ASCII	American Standard Code for Information Interchange
DES	Discrete Event Simulation
ID	Identifikátor
iMANET	Internet Based Mobile ad-hoc Networks
InVANET	Intelligent Vehicular ad-hoc Networks
MANET	Mobile ad-hoc Networks
OM	OPNET Modeler
RGB	Red Green Blue
VANET	Vehicular ad-hoc Networks
Wi-Fi	Wireless Fidelity

Zoznam príloh

Prílohy

Príloha A – Zdrojový kód funkcie v MATLABe pre generovanie trajektórie

```
function [x_coord,y_coord,pocet] =
zapis_variable_func(Xp,Yp,Pkrok,X0,Y0,Hpoc,Rpol,Hatr,Id);
clc;
clf ('reset');
%inicializacia premennych
% i je pocitadlo pre pocet krokov
i = 1;      % pocitadlo
smer=0;     % premenna s nastavenim smeru pohybu, 8 smerov + 1 nulovy pohyb
% bod startu stanice x y , alt je vyska , neuvazuje sa preto 0
x_coord = 0;
y_coord = 0;
%po vynulovani sa vlozia definovane hodnoty
x_coord(1) = Xp;
y_coord(1) = Yp;

alt = 0;
pocet=0;
%HOTSPOT premenne
% suradnice hotspotov
for ppp = 1:Hpoc
x0_coord(ppp) = X0(ppp);
y0_coord(ppp) = Y0(ppp);

end

hotspot_index = 1;          %index hotspotu
pocet_hotspotov = Hpoc;
R = Rpol;                      %polomer kruhu dosiahnutia hotspotu
Hotspot_attracting = Hatr;     %tag zapnutia hotspotov, 0 vyp, 1 zap
DIF_X = 0;                  %premenne pre vypocet kruhu okolo hotspotu
DIF_Y = 0;
%pocet skokov po mape
pocet_krokov=1;
pocet_krokov=Pkrok;
%velkost kroku
krok = 5;
matica = nacitanie_obr(krok,Id);       % vytvorenie matice s pst z bmp obrazka
velkost_obr = size(matica);     %zistenie velkosti obrazka pre nastavenie suradnic
mapy
%definicia smerovych matic
mat_U  = [0.03,0.79,0.03 ; 0.03,0.03,0.03 ; 0,0,0];
mat_D = [mat_U(3,3),mat_U(3,2),mat_U(3,1);mat_U(2,3),mat_U(2,2),mat_U(2,1);...
          mat_U(1,3),mat_U(1,2),mat_U(1,1)];
mat_L = [mat_U(1,3),mat_U(2,3),mat_U(3,3);mat_U(1,2),mat_U(2,2),mat_U(3,2);...
          mat_U(1,1),mat_U(2,1),mat_U(3,1)];
mat_R = [mat_U(3,1),mat_U(2,1),mat_U(1,1);mat_U(3,2),mat_U(2,2),mat_U(1,2);...
          mat_U(3,3),mat_U(2,3),mat_U(1,3)];
mat_UL = [mat_U(1,2),mat_U(1,3),mat_U(2,3);mat_U(1,1),mat_U(2,2),mat_U(3,3);...
          mat_U(2,1),mat_U(3,1),mat_U(3,2)];
mat_UR = [mat_U(2,1),mat_U(1,1),mat_U(1,2);mat_U(3,1),mat_U(2,2),mat_U(1,3);...
          mat_U(3,2),mat_U(3,3),mat_U(2,3)];
mat_DL = [mat_U(2,3),mat_U(3,3),mat_U(3,2);mat_U(1,3),mat_U(2,2),mat_U(3,1);...
          mat_U(1,2),mat_U(1,1),mat_U(2,1)];
mat_DR = [mat_U(3,2),mat_U(3,1),mat_U(2,1);mat_U(3,3),mat_U(2,2),mat_U(1,1);...
          mat_U(2,3),mat_U(1,3),mat_U(1,2)];
mat_S  = [0.1,0.1,0.1 ; 0.1,0.2,0.1 ; 0.1,0.1,0.1];
%suradnice mapy, potrebne pre zamedzenie pohybu mimo mapy
Mapa_X1 = 1+krok;
Mapa_X2 = velkost_obr(2)+krok;
```

```
Mapa_Y1 = 1+krok;
Mapa_Y2 = velkost_obr(1)+krok;
%otvorenie resp. vytvorenie noveho suboru *.trj pre zapis
fid = fopen('C:/trajectory_variable.trj','w+');
% vytvorenie hlavicky suboru *.trj
fprintf(fid,'Version: 4\n');                    %v opnet v16 nastavit na 3
fprintf(fid,'Position_Unit: Meters\n');     % jednotka suradnic
fprintf(fid,'Altitude_Unit: Meters\n');         % jednotka vysky
fprintf(fid,'Coordinate_Method: fixed\n');  % metoda nastavenia suradnic
fprintf(fid,'Altitude_Method: fixed\n');  % metoda nastavenia vysky
fprintf(fid,'locale: C\n');                     % povinny parameter
fprintf(fid,'Calendar_Start: unused\n');    % metoda nastavenia vysky
fprintf(fid,'Coordinate_Count: %d\n',pocet_krokov);  % pocet hodnot pohybu
%premenna matica do ktorej sa nacitava matica smeru
mat_SMERU = mat_S;
%testovanie zapnutia hotspotov a ich vykreslenie
    title(num2str(Id));
if Hotspot_attracting == 1;
   plot(x0_coord,y0_coord,'*','Color','r');
end
%opakuj pokial sa pocitadlo i nerovna poctu nastavenych krokov
while (i <= pocet_krokov)
       pocet=i;
%testovanie aktivneho hladania hotspotov
            if Hotspot_attracting==1
               DIF_X = x_coord(i) - x0_coord(hotspot_index);
               DIF_Y = y_coord(i) - y0_coord(hotspot_index);
%zistovanie pozicie vzhladom k hotspotu a nastavenie smerovej matice
%ak je hotspot nad aktualnou suradnicou Y
               if DIF_Y > 0
                  if DIF_X/DIF_Y <= -2
                     mat_SMERU = mat_R;
                  elseif (-2 < DIF_X/DIF_Y) && (DIF_X/DIF_Y <= -0.5)
                     mat_SMERU = mat_DR;
                  elseif (-0.5 < DIF_X/DIF_Y) && (DIF_X/DIF_Y <= 0.5)
                     mat_SMERU = mat_D;
                  elseif (0.5 < DIF_X/DIF_Y) && (DIF_X/DIF_Y <= 2)
                     mat_SMERU = mat_DL;
                  elseif  2 <= DIF_X/DIF_Y
                     mat_SMERU = mat_L;
                  end
%ak je hotspot pod aktualnou suradnicou Y
               elseif DIF_Y < 0
                  if DIF_X/DIF_Y < -2
                     mat_SMERU = mat_L;
                  elseif (-2 < DIF_X/DIF_Y) && (DIF_X/DIF_Y <= -0.5)
                     mat_SMERU = mat_UL;
                  elseif (-0.5 < DIF_X/DIF_Y) && (DIF_X/DIF_Y <= 0.5)
                     mat_SMERU = mat_U;
                  elseif (0.5 < DIF_X/DIF_Y) && (DIF_X/DIF_Y <= 2)
                     mat_SMERU = mat_UR;
                  elseif  2 <= DIF_X/DIF_Y
                     mat_SMERU = mat_R;
                  end
               elseif DIF_Y == 0
                   if DIF_X <= 0
                       mat_SMERU == mat_R;
                   elseif DIF_Y > 0
                       mat_SMERU == mat_L;
                   end
               end
            end
            r_smer = rand();    %generovanie nahoneho cisla pre zistovanie smeru zo
smerovej matice
            %vykreslenie cesty (-b) blue modrou farbou, sirka ciary 2
            plot(x_coord,y_coord,'-b','LineWidth',2);
            %vycitanie hodnot pst do matice vyseku z matice, v ktorej je spracovany
obrazok
```

```
            mat_Temp = [matica(y_coord(i)+2*krok,x_coord(i),1) ,
matica(y_coord(i)+2*krok,x_coord(i)+krok,1) ,
matica(y_coord(i)+2*krok,x_coord(i)+2*krok,1) ;...
                        matica(y_coord(i)+krok,x_coord(i),1) ,
matica(y_coord(i)+krok,x_coord(i)+krok,1) ,
matica(y_coord(i)+krok,x_coord(i)+2*krok,1) ;...
                        matica(y_coord(i),x_coord(i),1) ,
matica(y_coord(i),x_coord(i)+krok,1) , matica(y_coord(i),x_coord(i)+2*krok,1)];
            % vynasobena matica vyseku s maticou pohybu
            mat_Pom = mat_Temp.*mat_SMERU;
            %sucet pst vynasobenej matice
            suma = sum(sum(mat_Pom));
            % matica s absolutnymi hodnotami pst
            mat_Pom_abs = mat_Pom /suma;
            %suma = sum(sum(mat_Pom_abs))
            %vycitanie matice s indexami smerov do vektoru/matice
            perc_smer =
[mat_Pom_abs(1,1),mat_Pom_abs(1,2),mat_Pom_abs(1,3),mat_Pom_abs(2,1),mat_Pom_abs(2,
2),mat_Pom_abs(2,3),...
                         mat_Pom_abs(3,1),mat_Pom_abs(3,2),mat_Pom_abs(3,3);
5,1,6,3,9,4,7,2,8];
            %zistenie poctu nenulovych pst
                r = [0,0];
                pocet_nenul = 0;
                nenul_pom=1;
            for count = 1:9
                if ( perc_smer(1,count) > 0 )
                   pocet_nenul=pocet_nenul+1;
                end
            end
            %vycitanie nenulovych pst do vektoru/matice
            while nenul_pom<=pocet_nenul
                for count = 1:9
                    if (perc_smer(1,count) > 0)
                        perc_smer_nenul(1,nenul_pom)=perc_smer(1,count);
                        perc_smer_nenul(2,nenul_pom)=perc_smer(2,count);
                        nenul_pom = nenul_pom+1;
                    end
                end
            end
            pocet_nenul;
            nenul_pom;
            perc_smer_nenul;
            %pripravi vektor/maticu na testovanie pst a urcenie smeru
            for ii = 1:pocet_nenul
                r(1,ii+1)=r(1,ii)+perc_smer_nenul(1,ii);
                r(2,ii+1)=perc_smer_nenul(2,ii);
            end
            % urcenie smeru podla pst
            % vystup dsmer, smeru s identifikatorom cislo 1-9
            for iii=1:pocet_nenul
               if (r(1,iii) < r_smer) && (r_smer <= r(1,iii+1))
                  dsmer= r(2,iii+1);
               end
            end
            %testovanie smerov
             if (dsmer==1)        %ak sa ma vykonat smer hore      mat_U
                 % vycita do vektoru pst kazdeho najmensieho bodu medzi presunmi a
neskor zisti ci su vsetky nenulove
                 c_pom_pst = 0;
                 porad = 1;
                 for c = 1:1:krok
                     c_pom_pst(porad) =
(matica(y_coord(i)+krok+c,x_coord(i)+krok,1));     % ak neprechadzam cez bod s
nulovou pst
                     porad = porad+1;
                 end
```

```
                if (y_coord(i)+krok <= Mapa_Y2) &&
(matica(y_coord(i)+krok+krok,x_coord(i)+krok,1) > 0) && (all(c_pom_pst))       %ak
nejdem mimo mapy Y2
                                %nastav nove hodnoty a zapis ich do suboru
                                x_coord(i+1) = x_coord(i);
                                y_coord(i+1) = y_coord(i)+krok;
                                alt(i+1) = 0;
                                fprintf(fid, '%1.15f ,%1.15f ,%d ,0h0m5s ,0h0m0s ,0
,0 ,0 \n',x_coord(i), y_coord(i), alt(i));
                                mat_SMERU = mat_U;  % nastavim maticu taku aky bol
smer
                                %dsmer
                else
                    i=i-1;     % ak nevyhoveli X alebo Y suradnice
                end
            elseif (dsmer==2)     %ak sa ma vykonat smer dole      mat_D
                % vycita do vektoru pst kazdeho najmensieho bodu medzi presunmi a
neskor zisti ci su vsetky nenulove
                c_pom_pst = 0;
                porad = 1;
                for c = 1:1:krok
                    c_pom_pst(porad) = (matica(y_coord(i)+krok-
c,x_coord(i)+krok,1));     % ak neprechadzam cez bod s nulovou pst
                    porad = porad+1;
                end
                if (y_coord(i)-krok >= Mapa_Y1) && (matica(y_coord(i)+krok-
krok,x_coord(i)+krok,1) > 0) && (all(c_pom_pst))       %ak nejdem mimo mapy Y1
                                %nastav nove hodnoty a zapis ich do suboru
                                x_coord(i+1) = x_coord(i);
                                y_coord(i+1) = y_coord(i)-krok;
                                alt(i+1) = 0;
                                fprintf(fid, '%1.15f ,%1.15f ,%d ,0h0m5s ,0h0m0s ,0
,0 ,0 \n',x_coord(i), y_coord(i), alt(i));
                                mat_SMERU = mat_D;  % nastavim maticu taku aky bol
smer
                                %dsmer
                else
                    i=i-1;     % ak nevyhoveli X alebo Y suradnice
                end
            elseif (dsmer==3)     %ak sa ma vykonat smer dolava        mat_L
                % vycita do vektoru pst kazdeho najmensieho bodu medzi presunmi a
neskor zisti ci su vsetky nenulove
                c_pom_pst = 0;
                porad = 1;
                for c = 1:1:krok
                    c_pom_pst(porad) = (matica(y_coord(i)+krok,x_coord(i)+krok-
c,1));     % ak neprechadzam cez bod s nulovou pst
                    porad = porad+1;
                end
                if (x_coord(i)-krok >= Mapa_X1) &&
(matica(y_coord(i)+krok,x_coord(i)+krok-krok,1) > 0) && (all(c_pom_pst))     %ak
nejdem mimo mapy Y1,ak nejdem do pola s nulovou pst
                                %nastav nove hodnoty a zapis ich do suboru
                                x_coord(i+1) = x_coord(i)-krok;
                                y_coord(i+1) = y_coord(i);
                                alt(i+1) = 0;
                                fprintf(fid, '%1.15f ,%1.15f ,%d ,0h0m5s ,0h0m0s ,0
,0 ,0 \n',x_coord(i), y_coord(i), alt(i));
                                mat_SMERU = mat_L;  % nastavim maticu taku aky bol
smer
                               %dsmer
                else
                    i=i-1;     % ak nevyhoveli X alebo Y suradnice
                end
            elseif (dsmer==4)    %ak sa ma vykonat smer doprava        mat_R
                % vycita do vektoru pst kazdeho najmensieho bodu medzi presunmi a
neskor zisti ci su vsetky nenulove
                c_pom_pst = 0;
```

```
                porad = 1;
                for (c = 1:1:krok)
                     c_pom_pst(porad) =
(matica(y_coord(i)+krok,x_coord(i)+krok+c,1));     % ak neprechadzam cez bod s
nulovou pst
                     porad = porad+1;
                end
                if (x_coord(i)+krok <= Mapa_X2) &&
(matica(y_coord(i)+krok,x_coord(i)+krok+krok,1) > 0) && (all(c_pom_pst))      %ak
nejdem mimo mapy Y1
                                %nastav nove hodnoty a zapis ich do suboru
                                x_coord(i+1) = x_coord(i)+krok;
                                y_coord(i+1) = y_coord(i);
                                alt(i+1) = 0;
                                fprintf(fid, '%1.15f ,%1.15f ,%d ,0h0m5s ,0h0m0s ,0
,0 ,0 \n',x_coord(i), y_coord(i), alt(i));
                                mat_SMERU = mat_R;  % nastavim maticu taku aky bol
smer
                                %dsmer
                else
                    i=i-1;     % ak nevyhoveli X alebo Y suradnice
                end
            elseif (dsmer==5)    %ak sa ma vykonat smer hore dolava       mat_UL
                % vycita do vektoru pst kazdeho najmensieho bodu medzi presunmi a
neskor zisti ci su vsetky nenulove
                c_pom_pst = 0;
                porad = 1;
                for (c = 1:1:krok)
                     c_pom_pst(porad) = (matica(y_coord(i)+krok+c,x_coord(i)+krok-
c,1));     % ak neprechadzam cez bod s nulovou pst
                     porad = porad+1;
                end
                if (x_coord(i)-krok >= Mapa_X1) && (y_coord(i)+krok <= Mapa_Y2) &&
(matica(y_coord(i)+krok+krok,x_coord(i)+krok-krok,1) > 0) && (all(c_pom_pst))
                                %ak nejdem mimo mapy X1 a Y2
                                %nastav nove hodnoty a zapis ich do suboru
                                x_coord(i+1) = x_coord(i)-krok;
                                y_coord(i+1) = y_coord(i)+krok;
                                alt(i+1) = 0;
                                fprintf(fid, '%1.15f ,%1.15f ,%d ,0h0m5s ,0h0m0s ,0
,0 ,0 \n',x_coord(i), y_coord(i), alt(i));
                                mat_SMERU = mat_UL;  % nastavim maticu taku aky bol
smer
                                %dsmer
                else
                    i=i-1;     % ak nevyhoveli X alebo Y suradnice
                end
            elseif (dsmer==6)    %ak sa ma vykonat smer hore doprava      mat_UR
                % vycita do vektoru pst kazdeho najmensieho bodu medzi presunmi a
neskor zisti ci su vsetky nenulove
                c_pom_pst = 0;
                porad = 1;
                for (c = 1:1:krok)
                     c_pom_pst(porad) =
(matica(y_coord(i)+krok+c,x_coord(i)+krok+c,1));     % ak neprechadzam cez bod s
nulovou pst
                     porad = porad+1;
                end
                if (x_coord(i)+krok <= Mapa_X2) && (y_coord(i)+krok <= Mapa_Y2) &&
(matica(y_coord(i)+krok+krok,x_coord(i)+krok+krok,1) > 0) && (all(c_pom_pst))
                                %ak nejdem mimo mapy X1 a Y2
                                %nastav nove hodnoty a zapis ich do suboru
                                x_coord(i+1) = x_coord(i)+krok;
                                y_coord(i+1) = y_coord(i)+krok;
                                alt(i+1) = 0;
                                fprintf(fid, '%1.15f ,%1.15f ,%d ,0h0m5s ,0h0m0s ,0
,0 ,0 \n',x_coord(i), y_coord(i), alt(i));
```

```
                                mat_SMERU = mat_UR;  % nastavim maticu taku aky bol
smer
                                %dsmer
                    else
                        i=i-1;      % ak nevyhoveli X alebo Y suradnice
                    end
                elseif (dsmer==7)    %ak sa ma vykonat smer dole dolava      mat_DL
                    % vycita do vektoru pst kazdeho najmensieho bodu medzi presunmi a
neskor zisti ci su vsetky nenulove
                    c_pom_pst = 0;
                    porad = 1;
                    for (c = 1:1:krok)
                         c_pom_pst(porad) = (matica(y_coord(i)+krok-c,x_coord(i)+krok-
c,1));     % ak neprechadzam cez bod s nulovou pst
                         porad = porad+1;
                    end
                    if (x_coord(i)-krok >= Mapa_X1) && (y_coord(i)-krok >= Mapa_Y1) &&
(matica(y_coord(i)+krok-krok,x_coord(i)+krok-krok,1) > 0) && (all(c_pom_pst))
                                %ak nejdem mimo mapy X1 a Y2
                                %nastav nove hodnoty a zapis ich do suboru
                                x_coord(i+1) = x_coord(i)-krok;
                                y_coord(i+1) = y_coord(i)-krok;
                                alt(i+1) = 0;
                                fprintf(fid, '%1.15f ,%1.15f ,%d ,0h0m5s ,0h0m0s ,0
,0 ,0 \n',x_coord(i), y_coord(i), alt(i));
                                mat_SMERU = mat_DL;  % nastavim maticu taku aky bol
smer
                                %dsmer
                    else
                        i=i-1;      % ak nevyhoveli X alebo Y suradnice
                    end
                elseif (dsmer==8)     %ak sa ma vykonat smer dole doprava     mat_DR
                    % vycita do vektoru pst kazdeho najmensieho bodu medzi presunmi a
neskor zisti ci su vsetky nenulove
                    c_pom_pst = 0;
                    porad = 1;
                    for (c = 1:1:krok)
                         c_pom_pst(porad) = (matica(y_coord(i)+krok-
c,x_coord(i)+krok+c,1));     % ak neprechadzam cez bod s nulovou pst
                         porad = porad+1;
                    end
                    if (x_coord(i)+krok <= Mapa_X2) && (y_coord(i)-krok >= Mapa_Y1) &&
(matica(y_coord(i)+krok-krok,x_coord(i)+krok+krok,1) > 0) && (all(c_pom_pst))
                                %ak nejdem mimo mapy X1 a Y2
                                %nastav nove hodnoty a zapis ich do suboru
                                x_coord(i+1) = x_coord(i)+krok;
                                y_coord(i+1) = y_coord(i)-krok;
                                alt(i+1) = 0;
                                fprintf(fid, '%1.15f ,%1.15f ,%d ,0h0m5s ,0h0m0s ,0
,0 ,0 \n',x_coord(i), y_coord(i), alt(i));
                                mat_SMERU = mat_DR;  % nastavim maticu taku aky bol
smer
                                %dsmer
                    else
                        i=i-1;      % ak nevyhoveli X alebo Y suradnice
                    end
                elseif (dsmer==9)     %ak sa ma vykonat nulovy smer       mat_S

                                %nastav nove hodnoty a zapis ich do suboru
                                x_coord(i+1) = x_coord(i);
                                y_coord(i+1) = y_coord(i);
                                alt(i+1) = 0;
                                fprintf(fid, '%1.15f ,%1.15f ,%d ,0h0m5s ,0h0m0s ,0
,0 ,0 \n',x_coord(i), y_coord(i), alt(i));
                                mat_SMERU = mat_S;  % nastavim maticu taku aky bol
smer
                                %dsmer
                end
```

```
            %testovanie pozicie vzhladom k hotspotu, ak pozicia vyhovuje,
            %nastavi dalsi hotspot
            if ((DIF_X*DIF_X + DIF_Y*DIF_Y) < R*R) && (Hotspot_attracting == 1)
                hotspot_index = hotspot_index + 1;
                if hotspot_index == pocet_hotspotov + 1;   %ak nieje dalsi
hotspot, ukonci sa smycka pohybu
                i = pocet_krokov;
                end
            end
    i=i+1;
end
fclose('all');
```

Príloha B – Zdrojový kód procesu engine v OM

```
	//Ziskanie ID procesu
	 my_obj_id = op_id_self();

	//Ziskanie ID rodice procesu
	parent_obj_id = op_topo_parent (my_obj_id);

	//ziskanie nazvu stanice
	//node_atr_obj_id = op_topo_child (parent_obj_id, OPC_OBJTYPE_GENERIC, 0);
	op_ima_obj_attr_get (parent_obj_id, "name", &name);

	printf("\n Nazov stanice: %s \n",name);

	//Ziskanie ID potomka rodica , ID atributu engine.Mobility engine
	op_ima_obj_attr_get_objid (parent_obj_id, "engine.Mobility engine",
&mobility_engine_obj_id);
	mobility_engine_obj_id = op_topo_child (mobility_engine_obj_id,
OPC_OBJTYPE_GENERIC, 0);

	//Ziskanie hodnot atributov a ich vlozenie do premenych
	op_ima_obj_attr_get (mobility_engine_obj_id, "X pociatocna", &xp);
	op_ima_obj_attr_get (mobility_engine_obj_id, "Y pociatocna", &yp);
	op_ima_obj_attr_get (mobility_engine_obj_id, "MAX pocet krokov", &PKROK);
	op_ima_obj_attr_get (mobility_engine_obj_id, "Pocet hotspotov", &HPOC);
	op_ima_obj_attr_get (mobility_engine_obj_id, "Polomer hotspotu", &RPOL);
	op_ima_obj_attr_get (mobility_engine_obj_id, "Hotspot atribut", &HATR);
	op_ima_obj_attr_get (mobility_engine_obj_id, "Cas skoku",
&mymap_cas[parent_obj_id]);

	//podpora pre ziskanie suradníc hotspotov inym sposobom ich zadavania
	//op_ima_obj_attr_get (mobility_engine_obj_id, "X hotspot", &xx_string);
	//op_ima_obj_attr_get (mobility_engine_obj_id, "Y hotspot", &yy_string);

	//ziskanie ID atributu XY hotspot a poctu jeho potomkov
	op_ima_obj_attr_get_objid (mobility_engine_obj_id, "XY hotspot",
&hot_mobility_engine_obj_id);
	num[0] = op_topo_child_count (hot_mobility_engine_obj_id,
OPC_OBJTYPE_GENERIC);
	printf(" Pocet hotspotov stanice %s je: %.f \n",name,num[0]);

	//smycka pre vycitanie hodnot suradnic hotspotov z kazdeho potomka atributu
XY hotspot
	numcount = 0;
	for(numcount;numcount<num[0];numcount++)
	{
	poz_hot_mobility_engine_obj_id = op_topo_child (hot_mobility_engine_obj_id,
OPC_OBJTYPE_GENERIC, numcount);
	op_ima_obj_attr_get (poz_hot_mobility_engine_obj_id, "X suradnica", &x_pom);
	op_ima_obj_attr_get (poz_hot_mobility_engine_obj_id, "Y suradnica", &y_pom);
	xx[numcount] = x_pom[0];
	yy[numcount] = y_pom[0];

	printf(" %d Hotspot - suradnica X stanice %s je: %.2f
\n",numcount+1,name,xx[numcount]);
	printf(" %d Hotspot - suradnica Y stanice %s je: %.2f
\n",numcount+1,name,yy[numcount]);
	}

	/*	//podpora pre ziskanie suradníc hotspotov inym sposobom ich zadavania
	//vycitanie pola char a jeho konverzia na double cisla
	xx[0] = strtod (&xx_string[0],&pEndx);
	yy[0] = strtod (&yy_string[0],&pEndy);

	for(i=1; i<= HPOC[1]; i++)
		{
```

```
            xx[i] = strtod (pEndx,&pEndx2);
            yy[i] = strtod (pEndy,&pEndy2);

            pEndx = pEndx2;
            pEndy = pEndy2;
            }      */

      // Start MATLAB engine
      if (!(ep = engOpenSingleUse(NULL,NULL,NULL))) {
            exit(-1);
            }

      //blok predavania hodnot z OPNETu do MATLABu
      //vytvorenie premennych pre predanie hodnot z OPNETu do MATLAB Engine

      //vstupna premenna XP obsahujuca pociatocnu X suradnicu stanice
      XP = mxCreateDoubleMatrix(1, 1, mxREAL);
    memcpy((char *) mxGetPr(XP),(double *) xp, sizeof(double));
      //vlozenie hodnot premennych z OPNET do MATLAB
      engPutVariable(ep, "A", XP);

      //vstupna premenna YP obsahujuca pociatocnu Y suradnicu stanice
      YP = mxCreateDoubleMatrix(1, 1, mxREAL);
      memcpy((char *) mxGetPr(YP),(double *) yp, sizeof(double));
      engPutVariable(ep, "B", YP);

      //vstupna premenna pkrok obsahujuca maximalny pocet krokov
      pkrok = mxCreateDoubleMatrix(1, 1, mxREAL);
    memcpy((char *) mxGetPr(pkrok),(double *) PKROK, sizeof(double));
      engPutVariable(ep, "C", pkrok);

      //vstupna premenna X0 obsahujuca X suradnice hotspotov
      X0 = mxCreateDoubleMatrix(1, HPOC[1], mxREAL);
      memcpy((char *) mxGetPr(X0),(double *) xx, HPOC[1]*sizeof(double));
      engPutVariable(ep, "D", X0);

      //vstupna premenna Y0 obsahujuca Y suradnice hotspotov
      Y0 = mxCreateDoubleMatrix(1, HPOC[1], mxREAL);
      memcpy((char *) mxGetPr(Y0),(double *) yy, HPOC[1]*sizeof(double));
      engPutVariable(ep, "E", Y0);

      //vstupna premenna hpoc obsahujuca pocet hotspotov
      hpoc = mxCreateDoubleMatrix(1, 1, mxREAL);
    memcpy((char *) mxGetPr(hpoc),(double *) HPOC, sizeof(double));
      engPutVariable(ep, "F", hpoc);

      //vstupna premenna rpol obsahujuca polomer dosiahnutia hotspotu
      rpol = mxCreateDoubleMatrix(1, 1, mxREAL);
      memcpy((char *) mxGetPr(rpol),(double *) RPOL, sizeof(double));
      engPutVariable(ep, "G", rpol);

      //vstupna premenna hatr obsahujuca atribut zapnutia vyhladavania hotspotov (0
vyp,1 zap)
      hatr = mxCreateDoubleMatrix(1, 1, mxREAL);
      memcpy((char *) mxGetPr(hatr),(double *) HATR, sizeof(double));
      engPutVariable(ep, "H", hatr);

      //pretypovanie ID stanice na typ double
      doub = (double ) parent_obj_id;
      id[0] = doub;
      printf(" ID stanice %s je: %d \n",name,parent_obj_id);
      // vstupna premenna ID obsahujuca ID stanice, ID sa vypise ako titulok do
okna s trajektoriou
      ID = mxCreateDoubleMatrix(1, 1, mxREAL);
      memcpy((char *) mxGetPr(ID),(double *) id, sizeof(double));
      engPutVariable(ep, "I", ID);

      //vykonanie prikazu v matlabe, volanie vytvorenej funkcie
```

```
      engEvalString(ep, "[X_COORD, Y_COORD, P] =
zapis_variable_func(A,B,C,D,E,F,G,H,I)");
      printf (" Vygeneroval sa subor trj a suradnice \n");

      //buffer[BUFSIZE] = '\0';
      //engOutputBuffer(ep, buffer, BUFSIZE);
      //engEvalString(ep, "whos");

      //blok predavania hodnot z MATLABu do OPNETu
      //ziskanie vystupnych premennych z vytvorenej funkcie po jej vykonani
      XS = engGetVariable(ep, "X_COORD");
   YS = engGetVariable(ep, "Y_COORD");
      POC = engGetVariable(ep, "P");

      //vlozenie hodnot z premennych MATLABu do premennych OPNETu
      x_coord = mxGetPr(XS);
   y_coord = mxGetPr(YS);
      pocet = mxGetPr(POC);

      //ulozenie hodnot do mapy s rozlisovanim podla ID stanice
      mymapX[parent_obj_id] = x_coord;
      mymapY[parent_obj_id] = y_coord;
      mymapP[parent_obj_id] = *pocet;
      mymapCount[parent_obj_id] = 0;

      //engClose(ep);

      //vypis informacii do konzole
      printf(" Pociatocne X suradnice: %.4f \n",x_coord[count]);
      printf(" Pociatocne Y suradnice: %.4f \n",y_coord[count]);
      printf(" Celkovy pocet suradnic: %.4f \n",pocet[count]);

      //nastavenie premennych pre riadenie smycky nastavovania suradnic stanice
      //a ukoncenia nastavovania
      set = true;
      end = false;

      //prerusenie pre skok do dalsieho stavu
      op_intrpt_schedule_self(op_sim_time(), parent_obj_id);
```

Príloha C – Zdrojový kód procesu nastav v OM

```
	//Ziskanie ID procesu
	my_obj_id = op_id_self();

	// Ziskani ID rodica procesu
	parent_obj_id = op_topo_parent (my_obj_id);

	//pozastavenie simulacie po definovanu dobu
	op_intrpt_schedule_self (op_sim_time() + mymap_cas[parent_obj_id] , 0);

	//Ziskanie ID procesu
	my_obj_id = op_id_self();

	// Ziskani ID rodica procesu
	parent_obj_id = op_topo_parent (my_obj_id);

	//ziskanie nazvu stanice
	op_ima_obj_attr_get (parent_obj_id, "name", &name);

	count = mymapCount[parent_obj_id];
	//testovanie podmienky, ci existuju nove suradnice
	if (count < mymapP[parent_obj_id]){
		//vlozenie aktualnych hodnot suradnic do pomocnych premenych
	x_pos = mymapX[parent_obj_id][count];
		y_pos = mymapY[parent_obj_id][count];

		// Nastavenie atributu pozicie rodicovskej stanice na definovane
hodnoty
		op_ima_obj_attr_set_dbl (parent_obj_id, "x position", x_pos);
		op_ima_obj_attr_set_dbl (parent_obj_id, "y position", y_pos);

		//vypis aktualnych suradnic a ich poradia do konzole
		printf("\n Nazov stanice: %s ID: %d \n",name,parent_obj_id);
		printf(" X suradnice: %.4f \n",x_pos);
		printf(" Y suradnice: %.4f \n",y_pos);
		printf(" Cislo skoku: %d \n",count+1);

		//zvysenie poradia aktualnych suradnic o 1
		count = count + 1;
		mymapCount[parent_obj_id] = count;
		}

		//ak nevyhovie podmienka existencie dalsich suradnic, ukonci sa proces
ich nastavovania
		else {
			set = false;
			end = true;
			printf("\n Nazov stanice: %s ID: %d \n",name,parent_obj_id);
			printf(" Koniec suradnic \n");
			}
```

Príloha D – Hlavičkový blok procesného modelu engine v OM

```
#include "mobility_support.h"
#include "opnet.h"
#include "stdio.h"
#include <string.h>
#include <windows.h>
#include <stdlib.h>
#include "engine.h"
#include <iostream>
#include <map>
using namespace std;

//premena enginu
#define BUFSIZE 256
Engine *ep;
char buffer[BUFSIZE+1];

//definicia map pre ulozenie hodnot podla ID stanice
map<int,double *> mymapX;
map<int,double *> mymapY;
map<int,double> mymapP;
map<int,int> mymapCount;

map<int,double> mymap_cas;
//premenne pre medziulozenie hodnot suradnic
double x_pos, y_pos;

// definicia C++ vstupnych premennych pre funkciu v MATLABe
double xp[1]={0};
double yp[1] = {0};
double PKROK[1] = {0};
double xx[500] = {0};
double yy[500] = {0};
double HPOC[1] = {0};
double RPOL[1] = {0};
double HATR[1] = {0};
double id[1] = {0};

//pomocne premenne pre atributy
double x_pom[1] = {0};
double y_pom[1] = {0};
double num[1] = {0};

// definicia MATLAB vstupnych premennych pre funkciu v MATLABe
mxArray *XP;
mxArray *YP;
mxArray *pkrok;
mxArray *X0;
mxArray *Y0;
mxArray *hpoc;
mxArray *rpol;
mxArray *hatr;
mxArray *ID;
// definicia MATLAB vystupnych premennych z funkcie v MATLABe
mxArray *XS=NULL;
mxArray *YS=NULL;
mxArray *POC=NULL;

// definicia C++ vytupnych premennych z funkcie v MATLABe
double *x_coord = NULL;
double *y_coord = NULL;
double *pocet = NULL;
```

```
// pomocna premenna pre zistovanie poctu nastavenych suradnic
int count=0;
// premenna pre smycku for
int numcount;

//premenne pre vycitanie atributov, pomocne premenne pre konverziu string na double
char * pEndx;
char * pEndy;
char * pEndx2;
char * pEndy2;

char xx_string[500];
char yy_string[500];
char name[500];

//pomocna pre pretypovanie ID stanice na typ double
double doub;
```